InterFormNG Manual

version 2014

InterFormNG

eForms & Intelligent Document Management

www.interform400.com

InterForm A/S
Hyldalvej 11
DK-9541 Suldrup
Denmark

tel: +45 98 37 80 17
fax: +45 98 37 87 99
e-mail: support@interform400.com
sales@interform400.com

web: www.interform400.com

Document NGE-2014
User Manual for InterFormNG version 2014.R37

Printed in Denmark 2015

No part of this guide may be photocopied or reproduced in any way, except where noted, without the written consent of Interform A/S.

All the names in this manual are fictional and any resemblance to existing people or firms is purely coincidental.

© Copyright Interform A/S 2015. All rights reserved.

AS/400, iSeries , i5 and 'System i' are registered trademarks of the IBM Corporation.
HP-PCL is a trademark of Hewlett Packard
All other trademarks mentioned in this document are registered trademarks of their respective owners.

Table of Contents

Table of Contents 4

What is new in the 2014 Versions 9
- Designer 9
- E-mail output 9
- PDF output 10
- PDF Security Module 10
- PCL output 10
- Workflow 10
- Web interface 10
- Conversion of Designs/Templates 11
- New Module: JD Edwards EnterpriseOne connector 11
- License code is now required 11

Getting Started 12
- Prerequisites 12
- Installation 12
  - Important upgrade Information: 12
  - Installing Java Runtime Environment 12
  - Installing Java Runtime Environment on Windows 12
  - Installing InterFormNG 13
- The license code 18
  - The Trail Licence 18
  - Obtain a license 18
  - Alternative way to retrieve technical information 19
  - Install a licence 19
- Installed plugins 20
- Modules for InterFormNG 21
  - The PDF Security Module 21
  - The TxtToXML Module 21
  - The JD Edwards Connector Module 21
- Configuring InterFormNG 21
  - Set the language 21
  - Set the measure unit 21
  - Background image 22
  - Revision log 22
- Problem solving 24
  - Unable to start the designer - corrupted license 24
  - Unable to start the services 24
  - Error printing <xxx> document: already printing 24

Configuring InterFormNG 25
  - Configuring via the web interface 25
  - Changing the web interface port number 25
- System Configuration 28
  - General Systems Settings 28
  - Setting the log level 28
  - User configuration 29
- Input Configuration 30
  - File System Configuration 30
  - Txt to XML Configuration 31

Database Configuration. 32
Network configuration. 32
Enterprise Integration Configuration. 32
Scheduled Trigger Configuration. 32
Socket Trigger Configuration. 33
Processing. 33
Transform ZPL to PDF Configuration. 33
GS1 Databar Configuration. 33
Standard Barcode Configuration. 34
Image Import Configuration. 34
PCL Configuration. 34
PDF Configuration. 34
ZPL Configuration. 34
Output Configurations. 35
Printer configuration. 35
Archive Configuration. 36
Email Configuration. 37

Starting the Designer. 39
Before starting the designer the first time. 39
Getting started 39
Opening an XML file for designing. 41
A “Hello World” example. 42

Import external resources. 46
Importing a new font. 46
Importing XML file(s). 47
Using multiple XML files - an example. 47
Importing another template. 50
Importing a new image. 52
Setting the image size. 52
Importing a Resource. 53
Importing a translation table. 54
Example of translation files. 55

Designing templates. 56
Viewing Input and/or Result window. 57
Using Styles in InterFormNG. 58
Text style. 58
Geometry style. 64
Barcode style. 64
Image Style. 66
Color style. 67
Page Style. 67
Label Style. 68
Locale Style. 68
Print Style. 69
Error Style. 70
The Design Elements. 71
Text Element. 71
Line Element. 76
Box Element. 76
Barcode Element. 76
Image Element. 77
Page Element. 77

Label Element. 78
Group Element. 79
Include Element. 79
Fixed Area. 79
Flow Area. 81
New line Element. 82
ColumnBreak Element. 82
PageBreak Element. 82
Link Element. 82
ForEach Element. 86
Repeat Element. 86
Define Cursor Element. 87
Select Cursor Element. 87
Offset. 87
Offset update. 88
Variable Element. 88
Variable Update Element. 88
Metadata Element. 89
Combining Styles and groups. 90
Other important icons and design settings. 92

Design examples and tips. 93
Reuse groups. 93
Create Fixed areas when possible. 93
The Flow area. 95
The Page Static group. 96
Solution to: Element unreachable. 97
There are errors in the template. 98

Setting up InterFormNG for automatic conversions. 99
Installing/Uninstalling the Services. 99
Configuring the Services. 99
Starting and stopping the Services. 100
Using the Document Processor Service. 101
Setting up automatic printing. 101
Setting up automatic PDF creation. 102
Setting up e-mailing. 102
Selecting the output via metadata. 103
General metadata. 103
Email metadata. 106
Print metadata. 106

The InterFormNG WorkFlow. 109
Enabling the Workflow. 109
Working with Template Selections. 110
Working with actions. 112
Setting meta data values. 112
Working with Output Configurations. 113
Example Rule setups. 114

Transforming and Splitting XML files. 115
Activating the transform function. 115
Adding actions for the transformed files. 118
Same action for all transformed files. 118
Different action for each transformed file. 118

Creating xml files in specific folders. 119
Introducing XSLT. 120
Advanced transforming. 120
Splitting the intro_demo.xml file. 121

Appendix A: Using XPath with InterFormNG - in short. 123
Using variables in Xpath expressions. 125

Appendix B: Edit printing setup in setup files. 127
Selecting the printer via XML. 128
Structure of printConfig.xml 128
Scripting. 136
Access to meta data. 137
Printing labels on Zebra printers. 137
Requirements. 137
Connecting and installing the Zebra printer. 138
Updating the Print Configuration. 138
Installing the ZPL rendering plugin. 139
ZPL rendering plugin configuration. 139

Appendix C: Socket based XML processing. 140
Input format. 140
Header keys. 140
Result format. 140
Initiating a connection - A Java example. 141
Plugin configuration. 141
A Groovy web server example. 142

Appendix D: Installing on other platforms. 145
Installing on Linux. 145
Installing on AIX. 146

Appendix E: Processing a Database. 148
The Database. 148
JDBC Driver. 148
Configure InterFormNG's Database plugin. 149
Testing. 149
Example: Transferring meta data values from a Database into NG. 152

Appendix F: The TxtToXML Module. 154

Appendix G: The PDF Security Module. 156
Encryption and password protection. 156
Adding a digital signature. 156
Prerequisites. 156
Getting a digital certificate file. 157
Retrieve the alias from the certificate file. 160
Install Java Cryptography Extension (JCE). 160
Signing a PDF file. 161
Trusting signatures. 162
Verifying Digital Signatures. 164

Appendix H: Conversion of pre 2014 Templates. 165
Testing templates when upgrading to a 2014 Version or newer. 165
Conversion tool. 165
Problem when opening templates in pre 2014 Versions. 167

Using unconverted templates. 167

Appendix I: Using a Message Queue for input. 168
Enabling message queue in InterFormNG. 168
External message queue. 168
Sending data to message queue. 169

Index. 171

What is new in the 2014 Versions

This manual describes the functionality of the InterFormNG 2014 versions. Below are the additions/changes since the last official InterFormNG 2013 version.

Designer

- **Details/settings** of text elements are **shown in the element list** as a manually added description would do in the previous versions. This helps to get a better overview of what the individual elements do. This is described on page 71.
- Design **elements in error** are shown with a **red dot** (instead of a blue) to indicate a problem. (An example is shown on page 98)
- You will get a **warning when saving** a template if there are one or more errors inside the template. This is described on page 98.
- An **ELSE** option has been added for the IF element. Elements placed in the ELSE folder will be executed if the condition in the IF is not true. The ELSE option is described on page 85.
- The **Text style** has been enhanced to make it possible to align the text - even if the text is set to a specific width. The new parameter is called **Width adjustment** and is described on page 58. The parameters of the text style has been rearranged and this is also described on this page.
- A new function, **PAGE_NUMBER_TOTAL** (which outputs the total number of pages) is added. This and the other functions are described on page 72.
- You do no longer need to remember the various function names (e.g. PAGE_NUMBER_TOTAL). All supported **functions** are now selectable via a **drop down list.** This is shown on page 72.
- The definition of barcodes in the **barcode style has been changed**. This means, that existing customers may experience minor differences when comparing the barcodes to the result prior to the upgrade. The new barcode style is described on page 64. Please also note the remark about template conversion on page 11.
- Xpath **functions** supported are now not only Xpath 1.0, but also **Xpath 2.0** (Saxon home edition). This gives you even more possibilities for doing advanced calculations of e.g. the contents of the input XML files. **Appendix A** on page 123 lists some examples of how you can use Xpath.
- The **barcode element** can now **select the barcode type** itself. The settings for the barcode type is expected to be set with a prior barcode style. Previously you needed a barcode style each time you selected a different barcode type. The barcode element is described on page 76.

E-mail output

- The **e-mail** contents can not only be written as an .rtf file, but you can now also use an **html** file as well. This is covered on page 102.

PDF output

- The **texts and images** can now also **link to a URL**. The URL can be set not only as a constant, but e.g. also as an Xpath expression to make it possible to build up the link partially from information found in the XML file. Refer to page 71 for the text element and 77 for the image element, where this can be set.

- Easy interface for **archiving**. Now you can make InterFormNG generate an **XML file with index values** for **'real' archiving** e.g. in InterArchive. The XML file will be named the same as the output PDF file - if you are creating a PDF file. This option is described on page 74.

PDF Security Module

- A **new module** has been added: '**PDF Security**'. With this module you can both encrypt and password protect the output PDF files, limit the access to prevent users from printing, editing or copying the PDF contents and digitally sign the output PDF files. This new module is described in **Appendix G** on page 156.

PCL output

- It is now possible to **select the drawer (or tray)** from which the printer should take the paper in PCL. This feature is added on the page style element, which is described on page 67.

- **Support for black/white printers** (printers that does not support color commands) has now been added. The PCL output format is as default in color (which most B/W printers are able to print out), but you can now also set the output to be in black/white by setting this meta data key to false:
interform.destination.printer.pcl.image.fullcolor
The new meta data key is described on page 107.

- A new **meta data** key called: **multiup** can now be used for placing 2 pages of output on one page also called **N-up** or pages per side. For this function you should normally use the label element (as mentioned on page 78), but this meta data key can be used for creating a double sized output with 2 input pages - without changing the template design. This new meta data key is described on page 103.

Workflow

- The **split function**, that was previously added have been removed and exchanged for a **transform option in the workflow**. The transform is able to split xml files as before, but in a more simple way. The transform option is also able to transform xml files - hence the name. The transform feature is described on page 119.

- **Missing/no condition for a rule** now **selects all** xml files in the workflow. In the older versions of InterFormNG you needed a non-empty condition to select the files. This makes it easier to use the workflow. Refer to page 110 for details.

Web interface

- The default **port number** for the **web service** has been changed from **8080** (in the earlier versions) **into 1885.** If you are upgrading from an older version the old port

number will be preserved during the upgrade. Page 25 describes how to specify a specific port number for the web service.

Conversion of Designs/Templates

- When **upgrading** from a **pre 2014** version to a 2014 version (or newer) the templates will be converted into a new format. This has only importance if you are using barcode styles. If you are not using barcode styles you will not see any difference in the output and you can also open a template in any version of the designer no matter what version that created the template. If you are using barcode styles you cannot directly open a converted template in a pre 2014 version as the barcode styles are stored in a different manner. Refer to **Appendix H** on page 165 for details.

- A **template conversion** tool is started when you upgrade to a 2014 version - as described on page 17. The conversion can also be run after the upgrade as described on page 165.

New Module: JD Edwards EnterpriseOne connector

- Creates XML files from JD Edwards EnterpriseOne, which can be used as input in InterFormNG. Based on UBE (Universal Batch Engine) and handles standard and customized reports. The module is listed on page 21.

License code is now required

- In older versions you could use InterFormNG without a license code. This caused a 'TRIAL EDITION' text to be added to the output. Now a license code is required for the processor service and the designer. You can request a 7 days test license during installation/upgrade and via the web interface. This is described on page 14 and 18.

Getting Started

The idea of the following procedure is to give you a safe and easy start with the InterFormNG system. No knowledge on InterFormNG is required, and only minor Windows knowledge is needed. Knowledge of Xpath can be useful, but a short introduction is included in an appendix.

The main features of InterFormNG is, that it is able to transform XML files into PDF files or a printed result. The product is java based, so it is a platform independent application. During the transformation a design (including graphics, fonts, frames, lines etc.) is added. Here the installation and use is mainly shown when used with Microsoft Windows.

Prerequisites

- Java Runtime Environment 1.6 or newer must be installed. It is recommended always to use the latest update of Java Runtime Environment. We have e.g. found a problem in JRE version 1.6.0 Update 45, which had an negative effect on InterFormNG, which is solved simply by updating to the newest 1.6.0 update. JRE 1.8 and future JRE versions are not supported for 2014 versions of NG.

- Regarding accessing the windows machine via **Remote Desktop**: The user under which the InterFormNG processor service is running should not be used for signing on when accessing the machine on which InterFormNG is running. If it is necessary to do that, then you need to make sure, that no printers are shared on the client machine, that is accessing via remote desktop (or make sure, that there is not printer named the same as a shared printer on the client).

  (If printers are shared the printer setup in the InterFormNG processor may be affected and the error message **'Error processing print job: Error printing <xxx> document: already printing'** can be found in the service log file, while no print result can be found).

Installation

Important upgrade Information:

Please check out the important information in **Appendix H** on page 165 for the changes involved when upgrading from a pre 2014 version to a 2014 version or higher.

The section below describes installation on the Windows platform. Refer to **Appendix D** on page 145 for details of how to install on **Linux**. Installation on AIX is described on page 146. For installing InterFormNG on the **System i** platform the **InterXML** (iFormNG400) product is recommended (which includes the InterFormNG product).

Installing Java Runtime Environment

Before installing InterFormNG you need to install Java Runtime Environment version 1.6 or 1.7. You should use a newer version if possible. Currently you can download Java Runtime Environment (JRE) version 1.7 from here:

```
http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html
```

(Installing Java on windows require you to run the actual installer - not copying the JRE directory to the machine.)

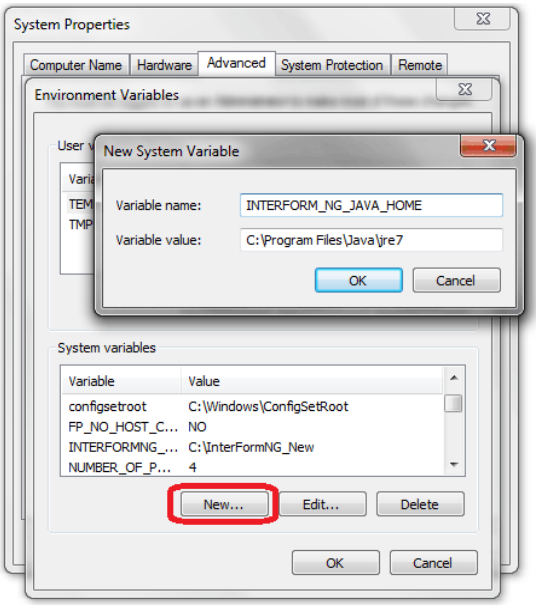
Installing Java Runtime Environment on Windows

If you are already using the InterForm400 graphical designer, then JRE is already installed.

If you already have installed a 32bit JRE on a 64 bit machine, then you need also to install a

64bit JRE and make sure, that InterFormNG is using the 64 bit JRE. That can be ensured by adding an enviroment variable called INTERFORM_NG_JAVA_HOME. The value for this should be the path to the Java library e.g. C:\Program Files\Java\jre7.

You can add/change an environment variable via **Control Panel**, **System** and **Advanced System settings** on the left. Now click **Environment variables..** in the bottom of the next window and then click **New..** under **System variables**:



Installing InterFormNG

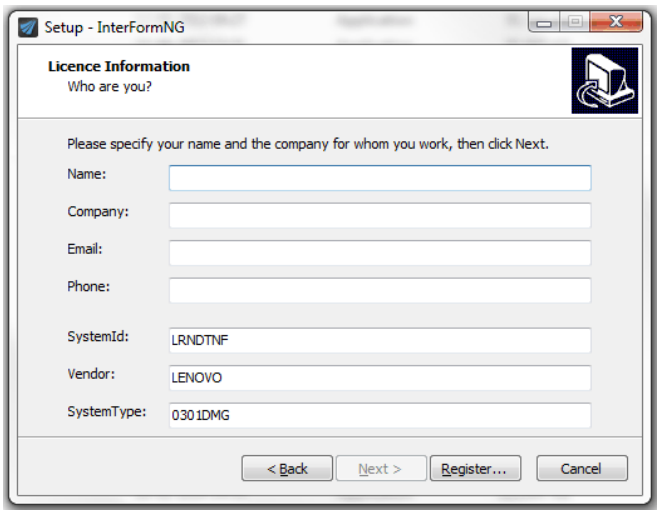
In order to install you first need to unzip the distribution file and then run the unzipped installation file. Now you will see these screens:

Setup - InterFormNG
Welcome to the InterFormNG Setup Wizard
This will install InterFormNG version 2014.R03 on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Intelligent output management
InterForm
Next >
Cancel

You need to accept the agreement in order to be able to proceed:

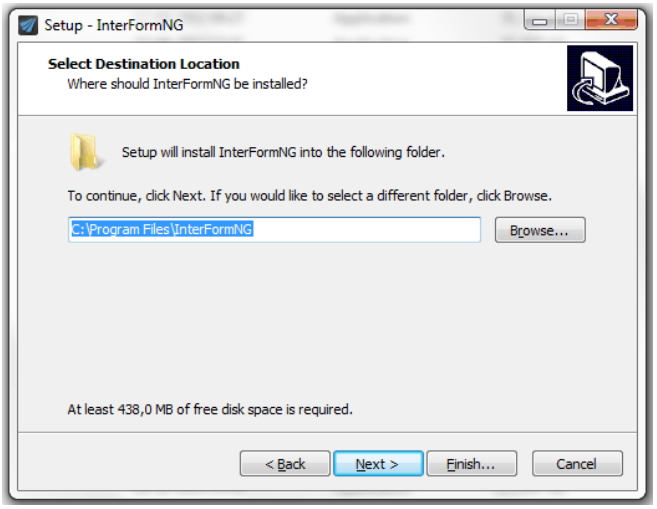
Setup - InterFormNG
License Agreement
Please read the following important information before continuing.
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
IMPORTANT: BY USING INTERFORMNG AND MODULES, YOU ARE AGREEING TO BE BOUND BY THE FOLLOWING INTERFORM A/S TERMS.
InterForm A/S (Licensor)
END USER LICENSE AGREEMENT
Concerning the licensing of the Software InterFormNG and modules (hereinafter referred to as "the Software").
Intellectual property rights
The Licensor holds the copyright and all other intellectual property rights to the
I accept the agreement
I do not accept the agreement
< Back
Next >
Cancel

If this is a first time installation (no licence code is found), then this screen is displayed:

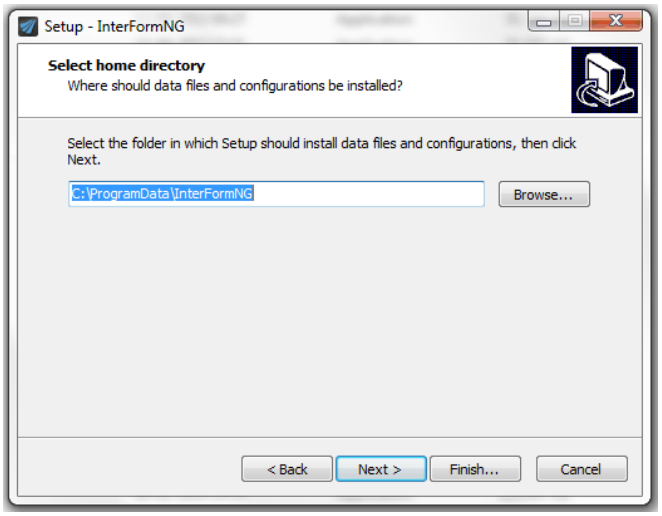


You need to register with some information about your PC in order to get a file with a license code. When you press Enter a contact is made to a licence server which will provide you with a free 7 day trial license. If this does not work you can later order a test code via the web interface as shown on page 18

Now state the place where InterFormNG should be installed. The path below is suggested:


Setup - InterFormNG
Select Destination Location
Where should InterFormNG be installed?
Setup will install InterFormNG into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\InterFormNG
Browse...
At least 438,0 MB of free disk space is required.
< Back
Next >
Finish...
Cancel

The next screen asks for the path for where data files and configuration files should be placed.

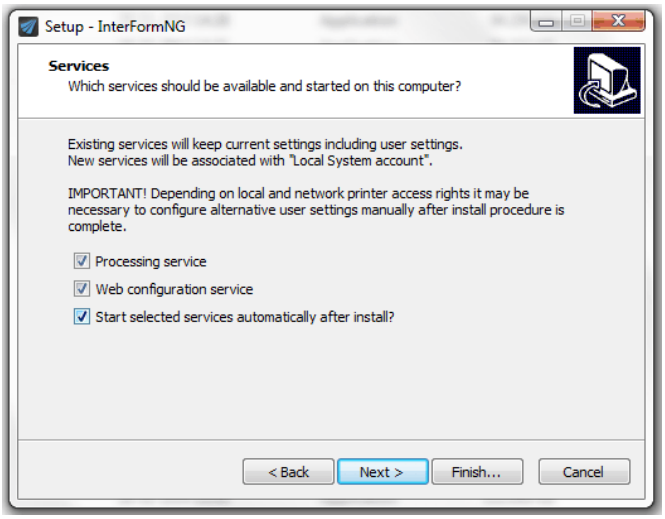

Setup - InterFormNG
Select home directory
Where should data files and configurations be installed?
Select the folder in which Setup should install data files and configurations, then click Next.
C:\ProgramData\InterFormNG
Browse...
< Back
Next >
Finish...
Cancel

The default path is:

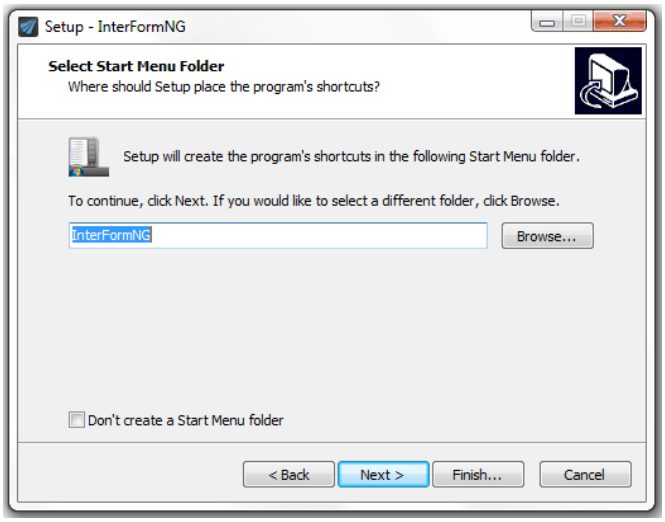
`C:\Program Data\InterFormNG`
(Please check/correct the exact path in your installation).

Please make sure if this path is OK and change it if needed. You will need to use this path whenever InterFormNG is exchanging files e.g. XML files, so a more simple path may be a good idea.

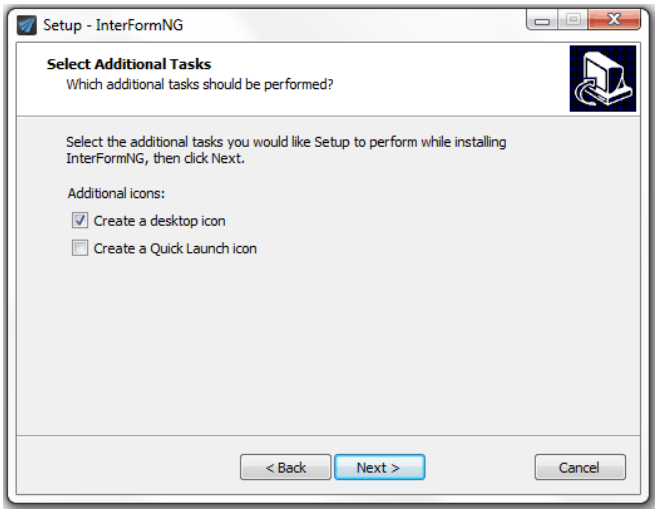
On the next screen you can decide which servers to install and if they should be started:
(Below I have chosen to start the services after install. As default they will **not** be started.).


Setup - InterFormNG
Services
Which services should be available and started on this computer?
Existing services will keep current settings including user settings.
New services will be associated with "Local System account".
IMPORTANT! Depending on local and network printer access rights it may be necessary to configure alternative user settings manually after install procedure is complete.
Processing service
Web configuration service
Start selected services automatically after install?
< Back
Next >
Finish...
Cancel

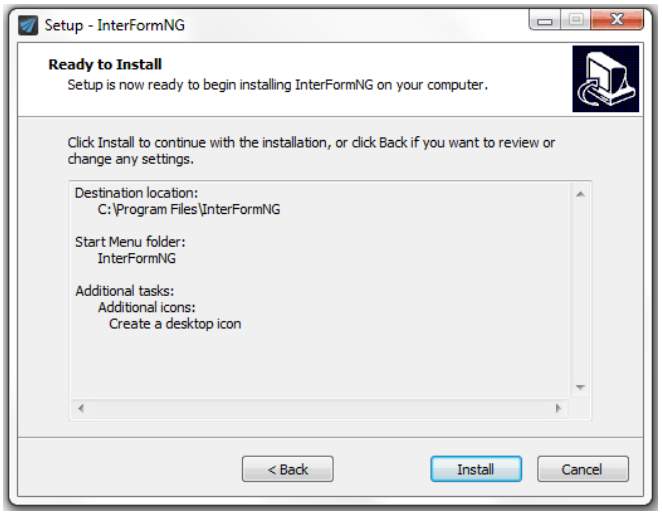
The next screen specifies where InterFormNG is placed in the Start menu folder. Alternatively you can chose not to create a Start Menu folder:


Setup - InterFormNG
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
InterFormNG
Browse...
Don't create a Start Menu folder
< Back
Next >
Finish...
Cancel

This screen asks if you want to create a desktop icon and a quick launch icon:

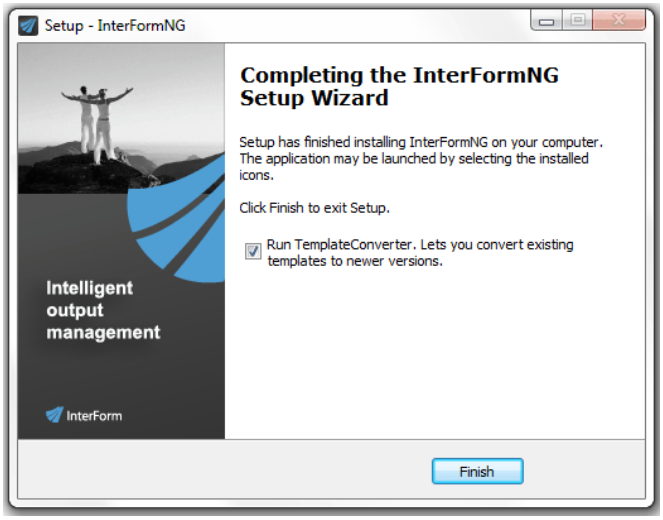

Setup - InterFormNG
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing InterFormNG, then click Next.
Additional icons:
Create a desktop icon
Create a Quick Launch icon
< Back
Next >
Cancel

The final screen shows you all the settings for the installation:


Setup - InterFormNG
Ready to Install
Setup is now ready to begin installing InterFormNG on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Program Files\InterFormNG
Start Menu folder:
InterFormNG
Additional tasks:
Additional icons:
Create a desktop icon
< Back
Install
Cancel

Now click 'Install' to start the installation.

After the installation the screen below is shown if the installation was actually an upgrade from a 2013 version (or earlier):


Setup - InterFormNG
Completing the InterFormNG Setup Wizard
Setup has finished installing InterFormNG on your computer. The application may be launched by selecting the installed icons.
Click Finish to exit Setup.
Run TemplateConverter. Lets you convert existing templates to newer versions.
Intelligent output management
InterForm
Finish

Unless you disable the 'Run TemplateConverter' option, then the convert tool will now start.

The license code

The Trail Licence

When you install and start InterFormNG for the first time, you can register and get a 7 day trial licence, which you can request during the installation. A trial licence has all features enabled. You can also request and install a licence via the web interface. InterFormNG will not run without a licence code.

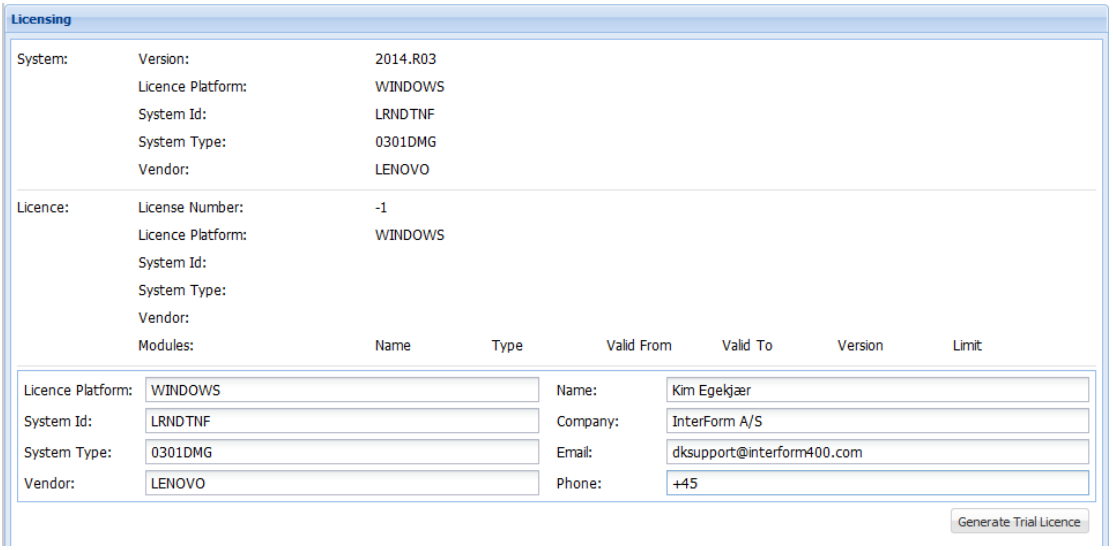
You can request a trial licence via the web interface if no licence file is found. This is described below.

Obtain a license

When you decide to order a licence to run InterFormNG, you must provide some information about your installation; either directly during installation, via the web interface or by email. Beside providing your personal details you should provide some technical information as well.

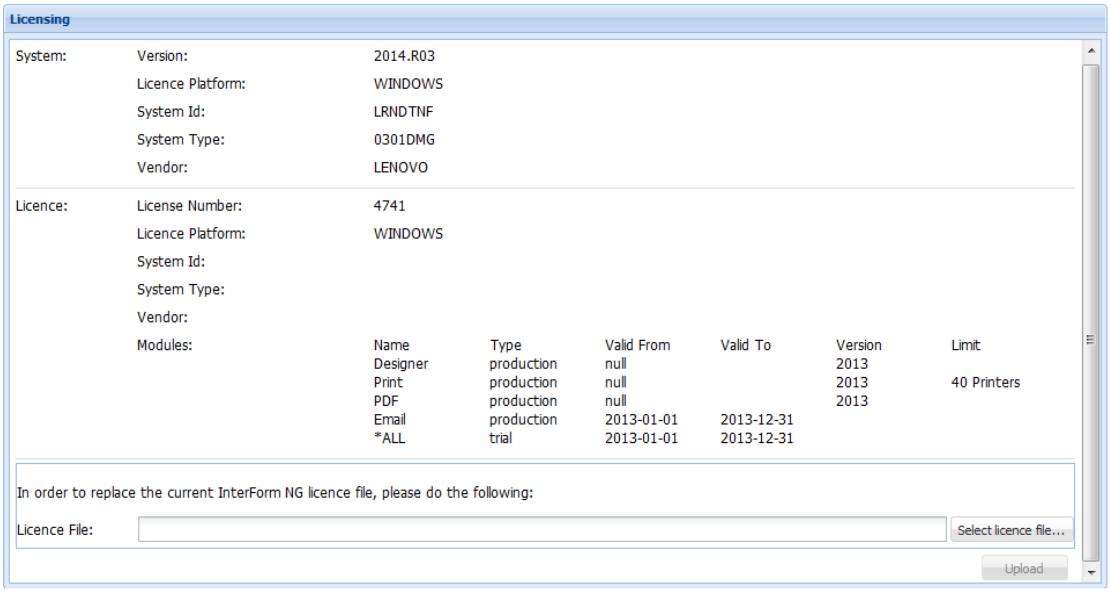
The simplest way to find this information is to install and start the (web) server. When accessing the web interface by selecting 'Start', 'All Programs' and run /InterFormNG/InterFormNG web admin'. Sign on e.g. as 'admin' with the password 'password' and then click 'System Configuration' followed by 'Licensing' to display the required information.

Then you will see a screen like below. Please copy and paste the characters (do not send a screen shot) to avoid any typos when the licence code is generated if you want to order a licence code via email.



You can install a licence code file by copying it directly into {INTERFORM_NG}\licence.

If you already have a licence code installed then the screen above looks like this:



Here you can load a new licence file if e.g. you want to try out an extra module or have purchased it. You simply click 'Select licence file..', select the file and press 'Upload' in the bottom right.

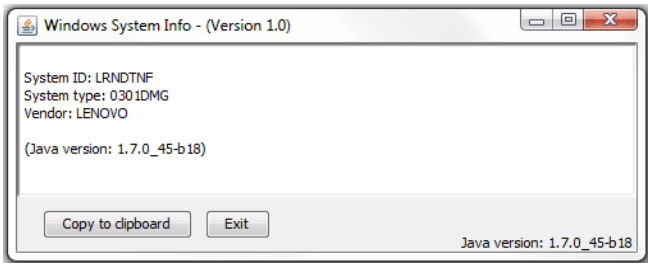
Alternative way to retrieve technical information

An alternative (for windows installations) is to run an .exe file which provides the information needed. The file can be used instead of installing InterFormNG to obtain the information. You can download the exe file via these links:

For 32bit windows machines:
http://download.interform400.com/shares/SW/WindowsSystemInfo32.exe

For 64 bit windows machines:
http://download.interform400.com/shares/SW/WindowsSystemInfo64.exe

When running the .exe file you will see a window like this:



Click the 'Copy to clipboard' icon and paste the text into an email (as text) to request a license code.

Install a licence

You can install a licence code via the web interface as shown above or by copying the licence file into the **licence** directory (a subdirectory to the InterFormNG home directory containing all settings and resources). You need to restart the designer and services to use the new licence file.

Installed plugins

The standard installation comes with these preinstalled plugins:

Plugin	Description
archive-manager	Archive documents to a folder
basic-printmanager	Process printed documents and send them to the correct printer
database-printjob	This plugin reads XML data from database records and converts it to PDF using an InterFormNG template
email-manager	Email document
filesystem-printjob	This plugin will scan a directory in the file system for XML files and print these files to PDF
gs1-databar-renderer	Render GS1 Databar family barcodes
pdf-renderer	Render XML documents into PDF format
standard-barcode-renderer	Render standard barcode types
transform-zpl-to-pdf	Transform ZPL script into readable PDF using an actual Zebra printer as a rendering engine
webadm-manager	Container for the administration web application
zpl-renderer	Render XML documents into ZPL (Zebra) format

Modules for InterFormNG

InterFormNG is delivered with a lot of functionality, but for those that want more, we also offer some modules, to enhance the product even more:

The PDF Security Module

If you purchase this module you can both encrypt and password protect the output PDF files, limit the access to prevent users from printing, editing or copying the PDF contents and digitally sign the output PDF files. This new module is described in **Appendix G** on page 156.

The TxtToXML Module

Apart from the core of InterFormNG it is also possible to purchase the TxtToXML module. The module makes it possible to use text files (e.g. csv files) as input in InterFormNG. They are then converted into XML before they are processed. Refer to **Appendix F** on page 154 for additional information.

The JD Edwards Connector Module

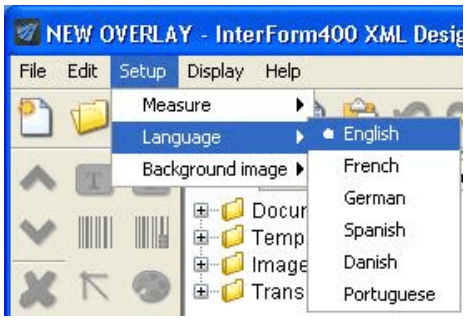
Creates XML files from JD Edwards EnterpriseOne. Based on UBE (Universal Batch Engine) and handles standard and customized reports. Please contact your local InterFormNG support for more information.

Configuring InterFormNG

After installing InterFormNG you should now start InterFormNG and configure a few things:

Set the language

If you want to change the language of InterFormNG you select this via 'Setup' and 'Language' as below:



A change of language will be active after restarting the designer. Currently only english is available.

Set the measure unit

The measure unit (when inserting layout elements) can be set to either of these:

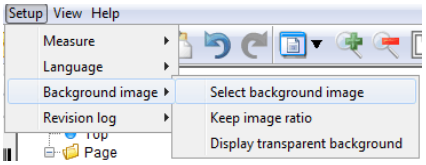
Inches and pels All measurements are typed in as inches as pels. Inches are entered before the comma (or decimal point) and pels are entered after. 240 pels equals 1 inch, so e.g. the value 1,120 equals one and a half inch.

Inches All measurements are entered in inches with a normal comma or decimal point. This means e.g. that the value 1,500 here equals one and a half inch.

CM All measurements are in centimeters with a comma or decimal point. This means e.g. that the value 3,81 equals one and a half inch.

Background image

It is possible to scan an image of a preprinted form and use this as a background to create a template with the same contents. To do this you select the image (bmp, jpg, pgn) via Setup and 'Select background image'and then select the image:



To enable the image you need to click on this icon on the lower left:
(This icon indicate that the background image has been enabled).

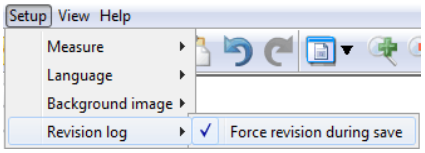
Possible options for the background image are:

Keep image ratio
The background image is scaled to fit the page - but still keeping the ration between height and width, so that the whole page will not be filled out (unless the ratio between height and width are the same for the page). Disable this to fit both the height and the width to the page i.e. the image will fill up the complete page.

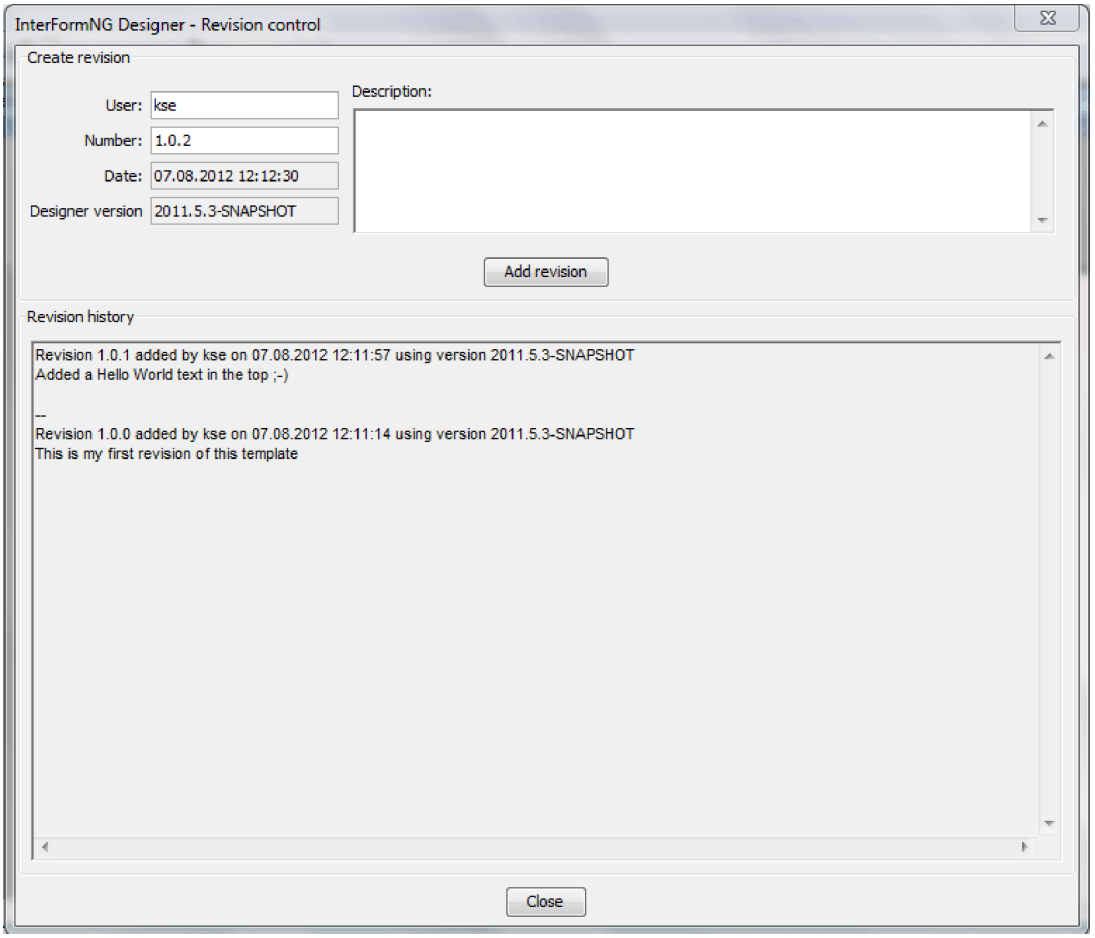
Display transparent background
You can make the background transparent to clearly see the new design/template. Enable this to do that.

Revision log

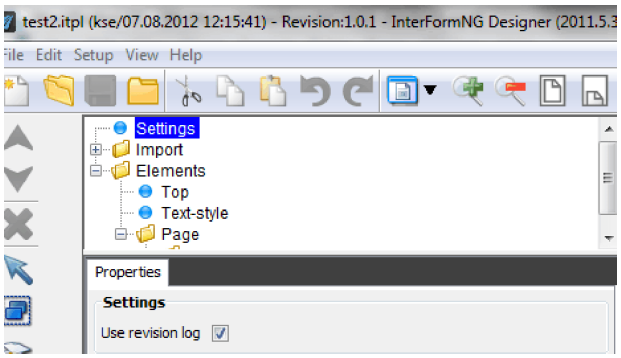
It is possible to activate a revision log on each template. When activated you are prompted for a description of the changes you have done (no journal of the changes is saved). Under Setup you can chose to activate a revision log for all templates, that you change by activating 'Force revision during save':



You can also set a revision log flag on each template, so any colleague (or customer) editing the same template will be prompted for a description and version number:


InterFormNG Designer - Revision control
Create revision
User: kse
Number: 1.0.2
Date: 07.08.2012 12:12:30
Designer version 2011.5.3-SNAPSHOT
Description:
Add revision
Revision history
Revision 1.0.1 added by kse on 07.08.2012 12:11:57 using version 2011.5.3-SNAPSHOT
Added a Hello World text in the top ;-)
--
Revision 1.0.0 added by kse on 07.08.2012 12:11:14 using version 2011.5.3-SNAPSHOT
This is my first revision of this template
Close

You can set the revision flag on the template here:



Problem solving

If you are unable to start the designer or the services stop with an error, then you could consult the list below:

Unable to start the designer - corrupted license

If you get this error you could try to right click the link for the designer and chose 'Run as administrator'. Alternatively you can chose to open up the authority of the InterFormNG directories - or chose to use another directory for the resources. On page 27 it is described how to change the authority to directories and files on windows.

Unable to start the services

If you are unable to start one or both of the services or if they are complaining of a corrupt license, then you can consider to open up authority to the used directories as described above - or chose to run the services under a specific user profile as shown on page 99.

Error printing <xxx> document: already printing

If you see the error message **'Error processing print job: Error printing <xxx> document: already printing'** in the service log file, while no print result can be found, then someone probably accessed the windows machine via **Remote Desktop**: The user under which the InterFormNG processor service is running should not be used for signing on when accessing the machine on which InterFormNG is running. If it is necessary to do that, then you need to make sure, that no printers are shared on the client machine, that is accessing via remote desktop (or make sure, that there is not printer named the same as a shared printer on the client).

The web service stops to work

Take care if you clean up the C:\windows\temp diretory as InterFormNG has some work files here. InterFormNG creates a subdirectory inside this. If you delete this, then the web service will stop working with this error message in the log file: **File "/login.jsp" not found**. In the future the temp directory will be moved elsewhere. To solve the issue you should just restart the web service.

Configuring InterFormNG

When InterFormNG is to convert new XML files into PDF files, e-mails with an attached PDF file or a printed result, you will need to configure InterFormNG and the necessary plugins.

Below it is described how to setup InterFormNG to enable each of these outputs and other options. The setup can be done either via editing the setup files via e.g. Notepad or via the InterFormNG web interface. The setup file for the service is stored as: \InterFormNG\profiles\default\application.properties within the configuration directory. The setup files for the plugins are stored in the configuration directory under \InterFormNG\profiles\default\plugins\.

Remember to always restart the processor service to activate any changes of the configuration. The only area, where this is not needed is for the workflow, if you have activated the **Auto refresh Workflow** option.

Below we will use the web interface to edit the configuration when possible:

Configuring via the web interface

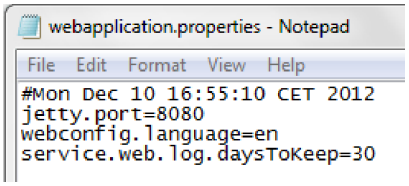
You can use the web interface to setup the plugins. You can start it in Windows by selecting 'Start', 'All programs', 'InterFormNG' and 'InterFormNG web admin'. A browser session will open and you will be asked for a User and Password. For the user you can state 'user' or 'admin' and you should state 'password' as the password. Please notice, that the password is case sensitive. If you logon as 'user' you can only view the settings; as 'admin' you can also edit the settings.

Changing the web interface port number

As default the web service will use this address: http://127.0.0.1:1885/webapp-springmvc
If you want to use a different port number you can change that in two ways:

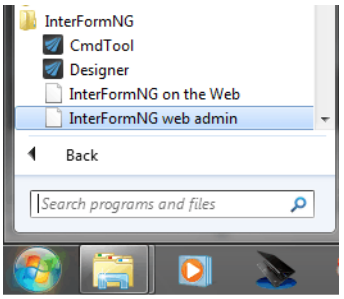
Via the **web interface**: You can set that as the Jetty port via General system settings as shown on page 28.

Via an **XML configuration file**. If you are unable to start the web server (e.g. because the port number is already used for something else), then you can change it by opening the xml configuration file inside {INTERFORMNG_HOME}/Profiles/default/webapplication.properties with Notepad and set the new port number to use as the Jetty port:

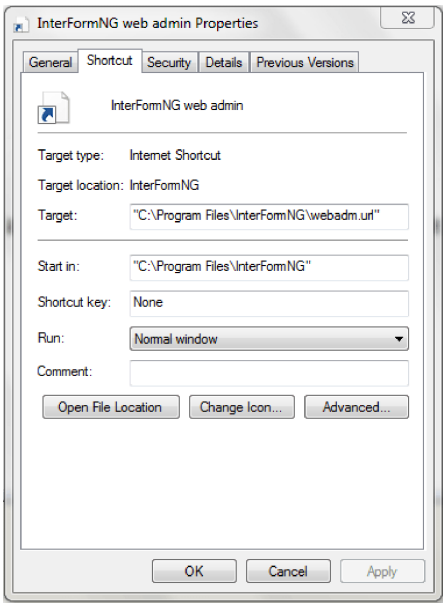


The changed port number will take effect from the next restart of the web server. Refer to page 100 for how to restart it.

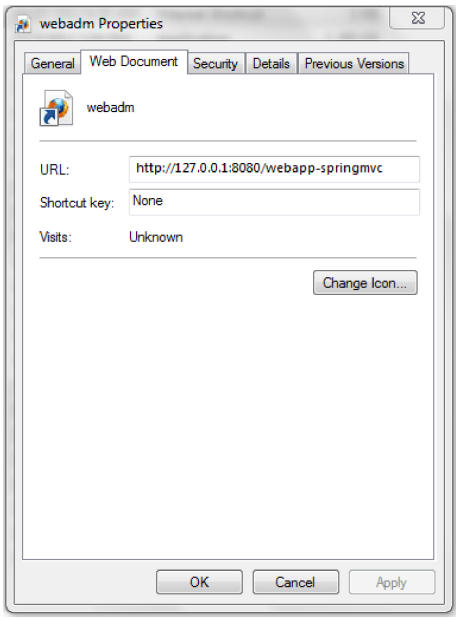
After changing the port number you might want to change the link you have installed for accessing the web interface:



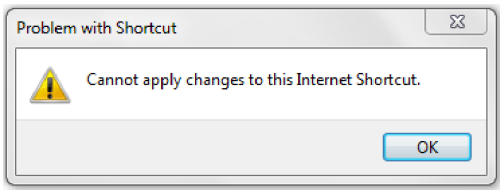
Now right-click on 'InterFormNG web admin' and click 'Properties' to find the file to change:



Now you should find the file and right-click it and select 'Properties' to change it:

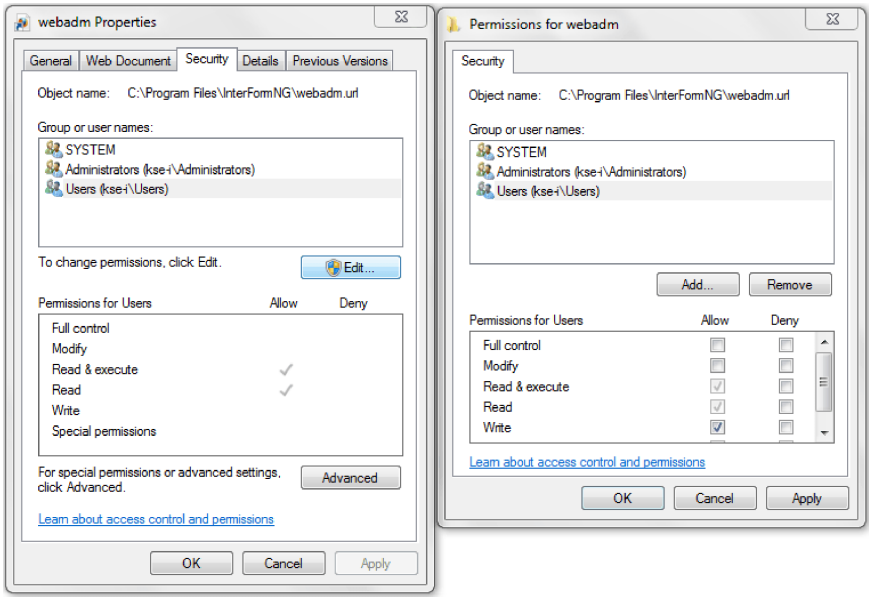


Now you can change the port number in the URL above. You might get an error message after the change when clicking 'OK':



Even though it is not clear, the problem is most likely, that you do not have authority to change the file. You can give yourself write authority in this manner:

On the Properties for the webadm file you click the 'Security' tab, select your user and click the 'Edit..' icon. This will open the window on the right. Here you activate 'Allow' for 'Write' and click OK:

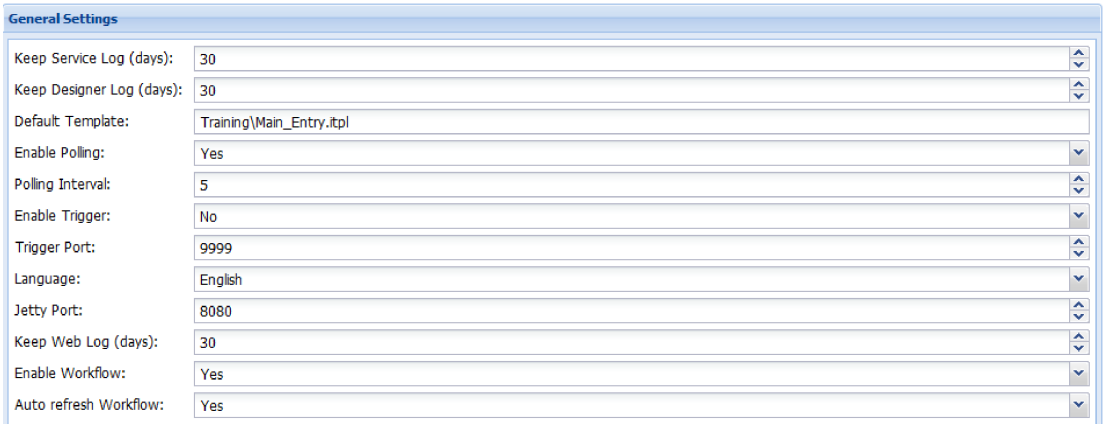


After the change you can click the 'Web Document' tab and change the port number.

System Configuration

General Systems Settings

Setting the default template and the polling interval is done on the screen below, which you find by clicking '**System Configurations**' and '**General Systems Settings**':



Keep Log (days)

InterFormNG creates log files for designer, the web server and the processing service inside {INTERFORMNG_HOME}/profiles/default/logs. Here you set for how long they should be kept.

Default Template

This is where you setup the template to use for all merges in the server. Normally this template should then include another other template depending on the contents of the XML file. This is not used if you are using the workflow, which is normally recommended.

Enable Polling

Select if you want the server to monitor an input directory via polling.

Polling Interval

The number of seconds between each time that the server will check for any new XML file in the monitored input directory.

Enable Trigger

If you want InterFormNG to run more efficient and react faster, then you should not use polling, but use a trigger instead. You enable the use of a trigger via this option.

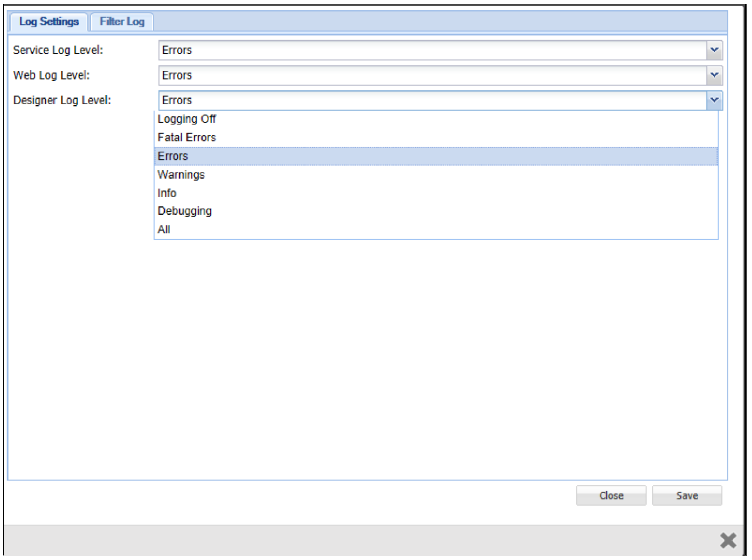
Trigger Port

If you enable the trigger as described above then this trigger port will be used. You signal that a new XML file is available by opening and closing a connection to this port. This technique works for both databases and files.

Return to the previous screen by clicking either 'Close' or 'Save' on the lower right (not shown above).

Setting the log level

To set the log level you logon to the web server and click '**System Configurations**' and '**Log Settings**':

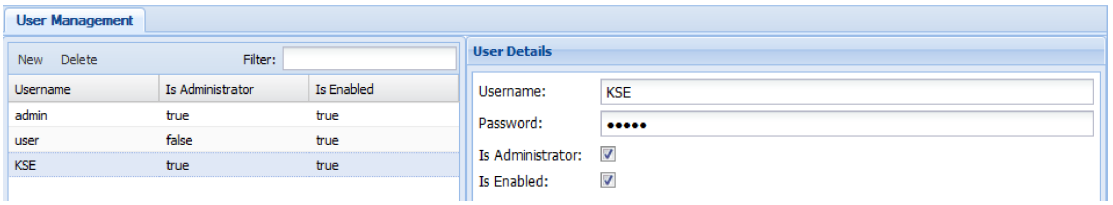


This is where you the level of details, that you would like to have included in the log files generated by the service, web server and the designer. You might be asked to change this to include more information in the log files in case of a support issue.

Notice that including a lot of details in the log files can have a negative impact on the performance of the product.

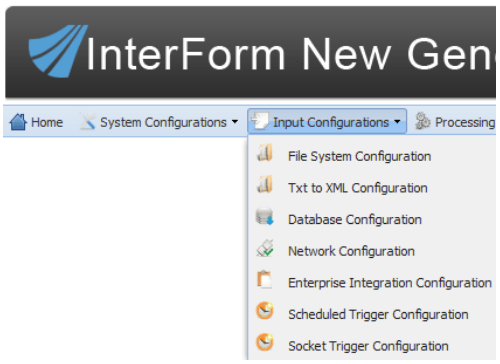
User configuration

As default InterFormNG is delivered with 2 users: Admin and user. The admin user is able to change anything, while the user is only able to display the setup. You reach this option from the web interface by selecting '**System Configurations'** and '**User Configuration**':



Input Configuration

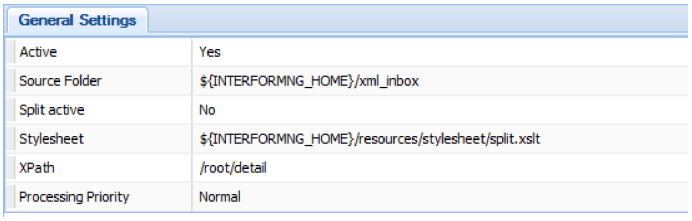
In the web interface you can setup the different input options by selecting 'Input Configurations':



Each option is described below.

File System Configuration

This is where you setup which directory to monitor for new XML files. This is setup by selecting '**Input Configuration**' and then '**File System Configuration**':



These are the configuration parameters:

Active Select 'Yes' to activate this plugin.

Source Folder

Here you can active the monitoring of new incoming XML files and state the directory to monitor for new files.

${INTERFORMNG_HOME} refers to the home directory, which is stored in the systems environment variable. Instead of using this you can also specify the complete path.

When used new subdirectories (in the monitored directory) will be created when needed:

/Processing The XML files that are currently being processed.

/Processed The XML files, that has successfully been processed.

/Errors XML files that could not be processed (an error occurred during processing). Look to the service.log file to find out what went wrong.

It is possible to monitor multiple directories. If you want to do that you simply separate each path with a comma like this:
${INTERFORMNG_HOME}/xml_inbox,Z:/xml_inbox,//192.168.250.250/apf3812home/xml_inbox
(Please notice, that we are here monitoring a directory on a different server, which is also possible).

Split active If you want to split up an XML file, then you should activate it via this option.

Stylesheet The path to the XSLT, which is used for the split. An XSLT file makes it possible to define a transformation from one XML file into one or more output XML files. Refer to page 115 for an in dept description and examples of how this can be defined.

Xpath This also refers to the split. You should define a path here. If InterFormNG finds that there are multiple branches with this path e.g. multiple detail nodes in the root in the example above, then InterFormNG will consider this to be an unsplitted XML file and use the XSLT to create new XML files.

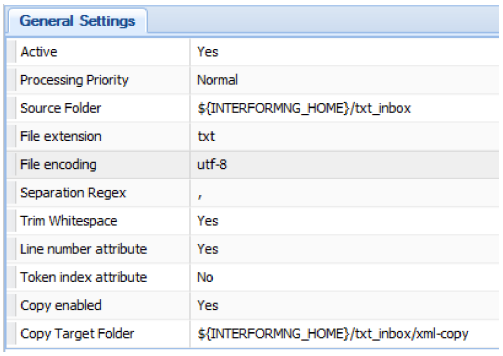
Please make sure to double check and test your split setup as you potentially can setup a loop, which will constantly create new XML files.

Processing Priority
Here you can set the priority of the file system plugin. You can set the priority to: Immediate, Normal or Batch.

Txt to XML Configuration

This is module for InterFormNG, so you cannot use it unless you have a license code which opens up for this functionality. This module makes it possible to convert text and csv files into xml files, which can be handled by InterFormNG.

The module has these configuration parameters:



General Settings

Active	Yes
Processing Priority	Normal
Source Folder	${INTERFORMNG_HOME}/txt_inbox
File extension	txt
File encoding	utf-8
Separation Regex	,
Trim Whitespace	Yes
Line number attribute	Yes
Token index attribute	No
Copy enabled	Yes
Copy Target Folder	${INTERFORMNG_HOME}/txt_inbox/xml-copy

The parameters specific for this module are:

Source Folder
The path to the folder, which is scanned for new text files.

File extension
The extension of the files, that you want to convert into XML.

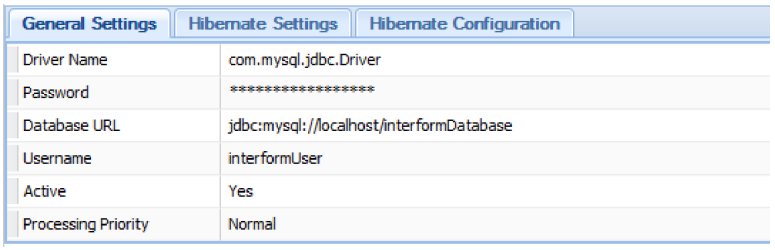
File encoding
The expected encoding of the incoming text files.

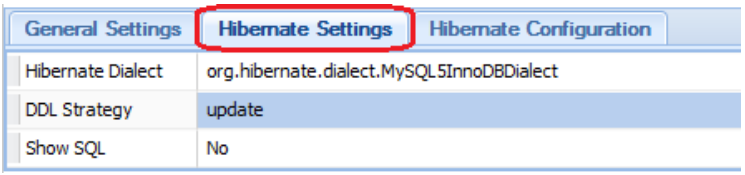
Separation Regex
The delimiter which defines the next element on each line in the text file. Each element will be inserted as a separate node in the XML file.

Trim Whitespace
If yes, any leading or trailing spaces after/before the delimiter will be removed before copying the file to XML.

Database Configuration

Instead of placing XML files in a monitored directory like above you can also place the XML file into a field in a database (while even setting one or more meta data keys) with the database plugin:

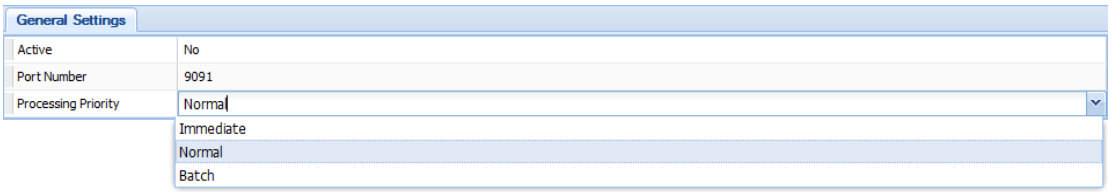




The database should be accessed via a JDBC driver. An example using a MYSQL database is included in Appendix D on page 148.

Network configuration

The network configuration sets some of the network properties for the Socket option, which is described later:

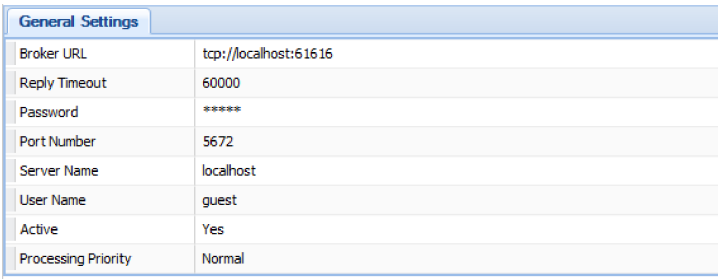


Here you enable the network processing and set the port number and priority used for socket requests.

As seen the possible priorities are: Immediate, Normal and Batch.

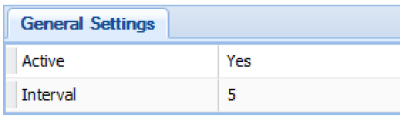
Enterprise Integration Configuration

With this you are using a message queue for telling InterFormNG when a new XML file is available:



Scheduled Trigger Configuration

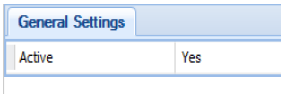
Use this option to activate the trigger e.g. every 5 seconds. With this you can gradually convert a polling setup into a setup using triggers by enabling this until you have fully implemented the trigger functionality.



Interval sets the number of seconds between each activation of the trigger.

Socket Trigger Configuration

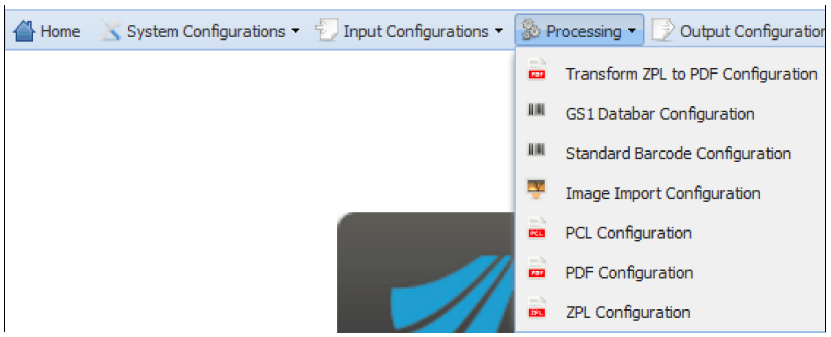
You can use the socket call to issue a time critical conversion in InterFormNG. You can e.g. a URL back linking to the resulting PDF file. You setup various properties in the Network Configuration above and you enable the socket function here:



The socket trigger is described in **Appendix C** on page **140**.

Processing

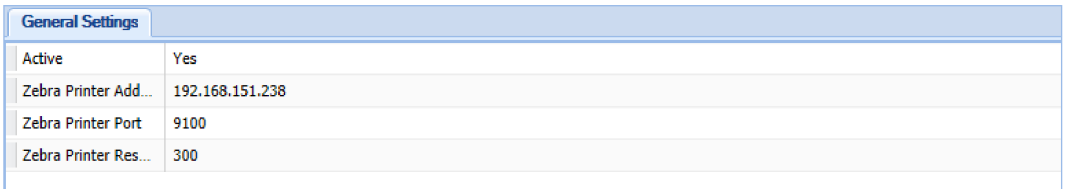
InterFormNG is able to generate multiple types of output. You can setup the properties for each under 'Processing' in the web interface:



Below each of the configurations are described:

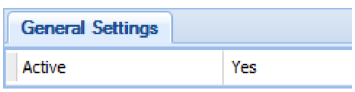
Transform ZPL to PDF Configuration

When designing a template for ZPL output you can view the output in the Result window, but if you want to see exactly how the final result would look, then you can setup InterFormNG to use a physical printer to generate an image of the output. Such a printer should not be used for production printing and should be standing idle for this to work. If you want to use this feature you select '**Processing**' and '**Transform ZPL to PDF Configuration**':



GS1 Databar Configuration

If you want to use GS1 barcodes you should activate the feature here:



Standard Barcode Configuration

If you want to use barcodes in InterFormNG, then this should be activated:

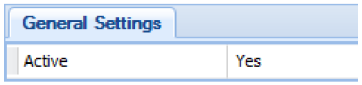
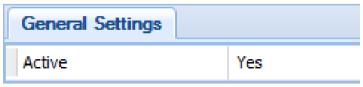


Image Import Configuration

If you want to use images in InterFormNG, then this should be activated:

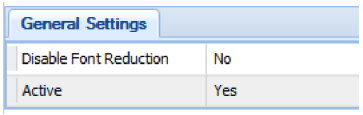


PCL Configuration

Not used.

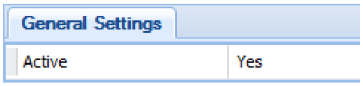
PDF Configuration

Here you activate the possibility to create PDF files. If you set the **Disable Font Reduction** feature to **Yes**, then the PDFs will be larger, but will be created a little bit faster.



ZPL Configuration

If you want to be able to print the ZPLII data stream, then this should be activated:

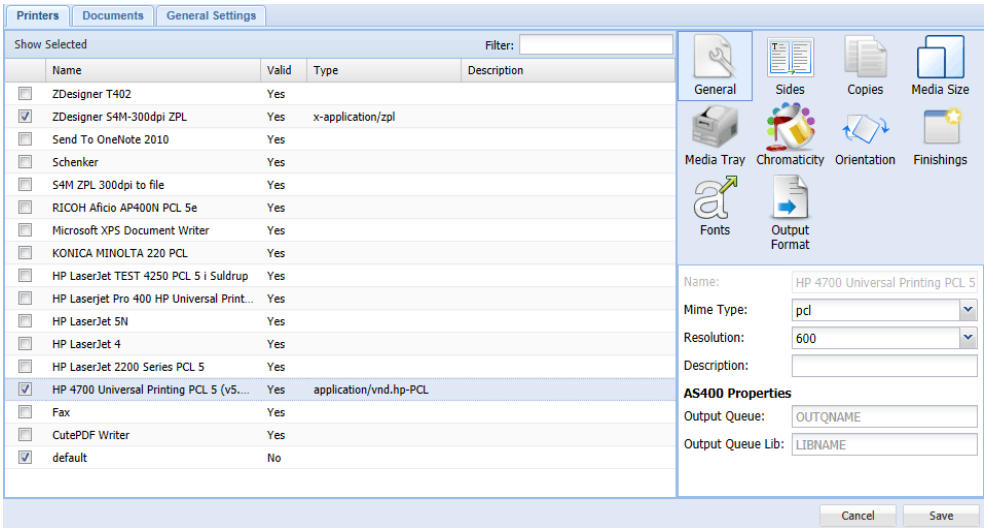


Output Configurations

Here you set the attributes for the outputs, that can be generated by InterFormNG:

Printer configuration

Under the 'Printers' tab you can change settings for the printers and activate/disable installed printers. Only activated printers can be used in InterFormNG. The main settings are these:



(The default printer refers to the default printer on the machine. If you deselect/deactivate this printer, then it will disappear).

Name The name of the printer. As default this will be the same as the windows printer.

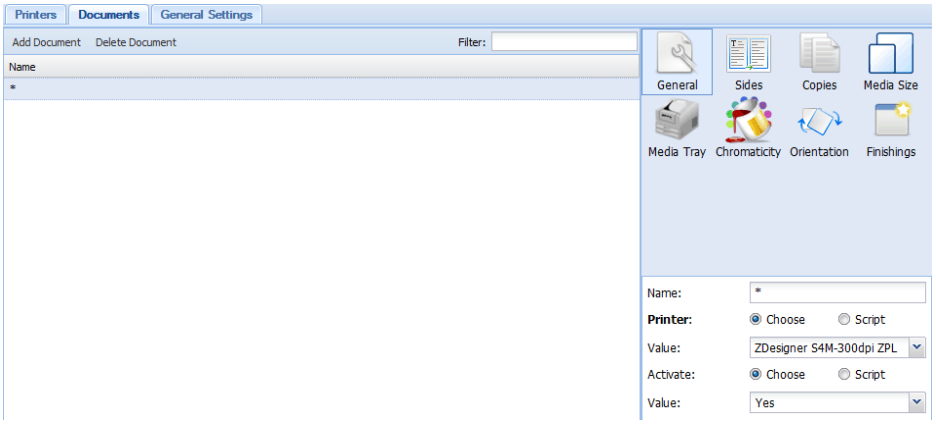
Mime Type The print data type, that will be sent to the printer. Choose between 'pcl', 'postscript' and 'zpl'. For normal laser printers pcl is recommended.

Resolution The printers resolution (in dots per inch). This is normally 600 for laser printers and either 200 or 300 for ZPL label printers.

By clicking on the icons on the left you can set a lot of other printer settings.

Documents (Printer selection)

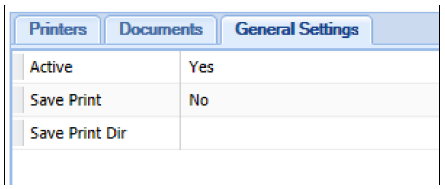
The printer can be selected directly in the used template by setting meta data key, 'interform.destination.printer.printerName' (selecting via the name), via 'interform.destination.printer.printerId ' (via the printer ID, which is found in the config.xml document) or better by setting the meta data key 'documentType' in the template and linking the documentType with a printer here. You get here by selecting '**Output Configurations**', '**Printer Configuration**' and the '**Documents**' tab in the top:



For the document type you can set a single one with the value of '*', if you want only to use one specific printer and then link this value with the printer in question. Alternatively you can add various document types e.g. 'invoice' and 'label' and select that as the meta data 'documentType' in the template.

Enable printing

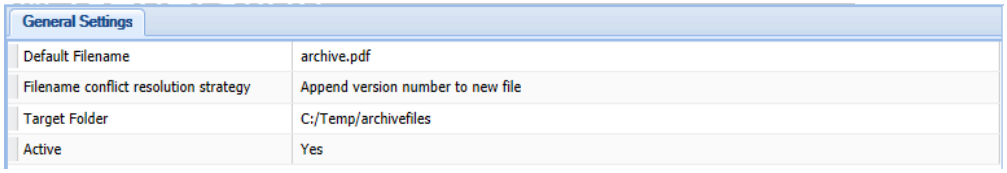
Before you can actual print you will need to enable printing. This is done by selecting '**Output Configurations**', '**Printer Configuration**' and the 'General Settings' tab in the top:



Set the 'Active' field to 'Yes' to enable printing.

Archive Configuration

If you want to generate PDF files, then you set it up here. You find this by selecting '**Output Configurations**' and then '**Archive Configuration**':



Default File Name

This is the file name, that will be created as default. If the file happens to exist, then a file named archive(1).pdf will be generated. For subsequent runs a new number is added (archive(2).pdf etc...). The Default File Name can be overridden in the template by setting a meta data key called 'interform.plugin.archiver.fileName'.

Filename conflict resolution strategy

Here you specify how InterFormNG should react if InterFormNG is trying to create a file when it already exists:

Rename existing file to include date: Renames the existing file with a time stamp and creates the new file with the requested name.

Append version number to new file: Creates files as e.g. default(1).pdf

Overwrite existing file: Overwrites the existing file

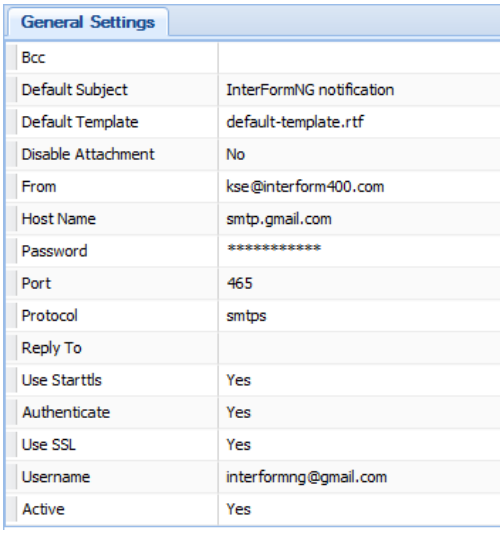
Target Folder

The target folder in which the PDF files will be created. This works with the meta data key 'interform.plugin.archiver.folderName', which can be defined within the template, so that the path written in this meta data key is the path within the target folder specified here. (You could think of the target folder as the root folder for the pdf/archive generator).

Active You need to set this to '**Yes**' to enable the possibility to create PDF files.

Email Configuration

By selecting '**Output Configurations**' and then '**E-mail Configuration**' you reach this screen:



General Settings

Bcc	
Default Subject	InterFormNG notification
Default Template	default-template.rtf
Disable Attachment	No
From	kse@interform400.com
Host Name	smtp.gmail.com
Password	**********
Port	465
Protocol	smtps
Reply To	
Use Starttls	Yes
Authenticate	Yes
Use SSL	Yes
Username	interformng@gmail.com
Active	Yes

The fields sets this:

BCC A BCC receiver.

Default subject

The subject of the outgoing e-mail.This can be overridden in the used template by setting a value for the meta data key '**email.subject**'.

Default Template

This refers to an e-mail template (an .rtf or html file) setting the contents of the e-mail. You can overridde this choice in the (form) template via the meta data key 'email.templateName' and here state a different file (email template).

Disable attachment

Normally the email will be sent out with a pdf file containing the result of the merge between a template and the XML file. This attachment can be disabled with this.

From The e-mail address specified as the sender.

Password Notice that the password is not shown as readable text, but is saved unencrypted in a configuration file.

Reply To The email address to reply to.

Active Set this to 'Yes' to enable e-mailing.

The rest of the fields are setup for the outgoing e-mail server. Above the settings for gmail are used. You should check with the administrator of your email server to find the settings, that you should use. Generally used port numbers are: 25, 465 and 578.

Port 25 is normally not using SSL, so for that you should try with these settings:
Protocol: **smtp**
Use SSL: ***No**

Hidden gems
When a program tries to send mail, it sends the command EHLO followed by its hostname. In some cases it only sends the hostname, but the mail server expect a fully qualified domain name (FQDN).

If my PC is named Egon-PC and is on the domain interform400.com, it should send the following command
EHLO Egon-Pc.interform400.com
But it only sends
EHLO Egon-Pc

To force it to send the FQDN you can add the line:

email.smtp.localhost=Egon-Pc.interform400.com

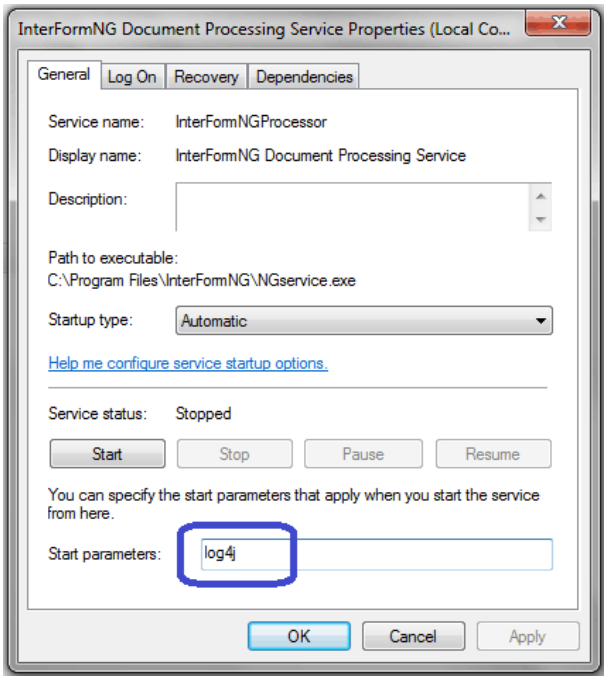
- to the **plugin.properties** file in the directory:
{INTERFORMNG_HOME}\InterFormNG\profiles\default\plugins\email-manager\conf

And restart the service.

To see what is sent to/from the mail server, you can turn debugging on by adding this line to the plugin.properties file as well:

email.debug=true

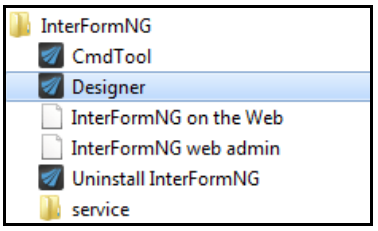
You also have to add the startparameter 'log4j' to the service:



You can now find this in the service.log file: EHLO Egon-Pc.interform400.com

Starting the Designer

If you have chosen the default path in the Start Menu you can start it by selecting 'InterFormNG', 'Designer' as below:



Notice: The templates are stored as .itpl files. You should not edit these files in any other way than the InterFormNG designer. Editing the files with another application may make the templates unusable.

Before starting the designer the first time

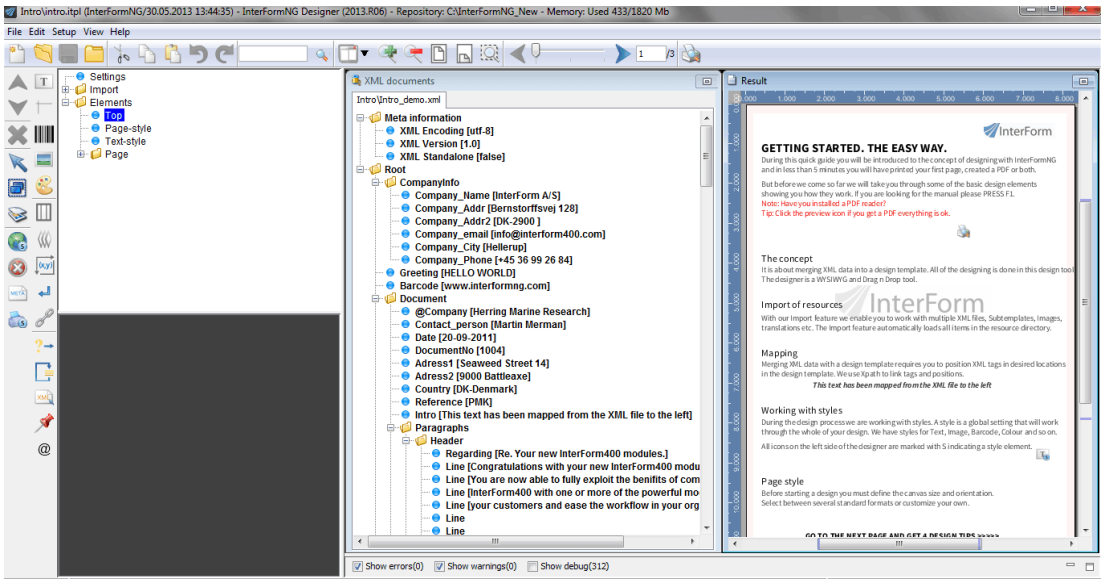
Before using the designer for the first time, you will probably want to copy one or more fonts to the resource directory. You can e.g. do that by copying True Type Fonts (TTF) from C:\Windows\Fonts into the Fonts directory in the InterFormNG Resource directory. That makes the fonts available for InterFormNG. Details of font import can be found on page 46.

If the designer was running while you copied the fonts, then you should restart the designer to make the fonts 'visible' for the designer.

The designer is shipped with some preinstalled resources (fonts, templates, images and xml files), so even if you do not copy any resources to the resource directory you can still start to work with the product 'out of the box'.

Getting started

The first time you start the designer you will see this:



This is a combined introduction and demo of the designer. The leftmost 'window' displays the list of design elements, the middle contains a demo xml file and the rightmost window displays the result. For a first time user it is recommended to read

the all the pages in the result view to get a fast introduction. You can click through the pages with the arrow icon on the top right:

the resources for this intro template are stored in subdirectories called ‘intro’.

Disabling the Intro template
The intro template is shown each time you open the designer - unless you select ‘Setup’ and ‘Demo options’ and remove the flag for ‘Open Demo template on startup’.

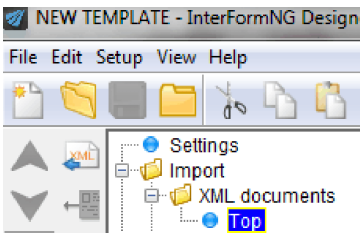
You can create a new template either by selecting ‘File’ and ‘New overlay’ in the upper menu or by clicking on this icon on the upper left:

Changing the settings for a new template
The new template is ‘born’ with predefined elements to help you to get starting. The basis for the new template is a template called ‘default.itpl’, so if you want new templates to open with other elements or different settings, then you can simply save it to a template called default.itpl. You can use this if you e.g. want to change the default page size (in the page style) into the letter size - as used in e.g. US.

If you want to open an existing template, you select ‘File’ and ‘Open’ or click this 2nd icon on the upper left of the screen:

Opening an XML file for designing

When you want to design create a template you should use it to redesign an XML file. By selecting the yellow ‘Import’ folder and then the subfolder ‘XML documents’ and the ‘Top’ inside you can refer to an XML file by clicking this icon on the left:

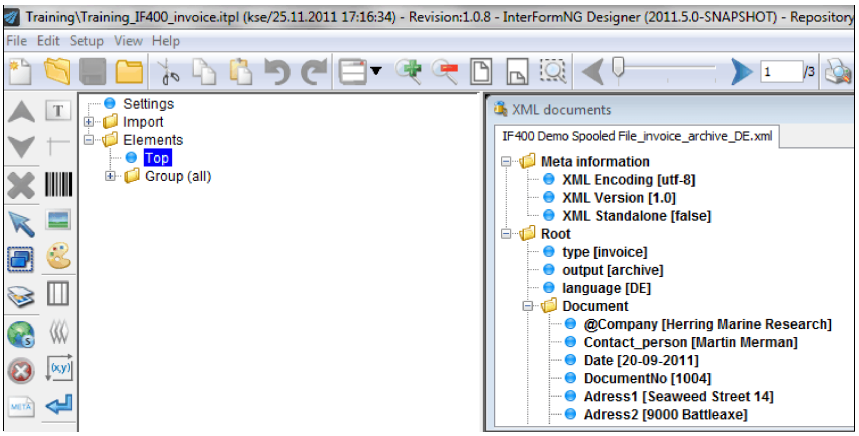


When you click the icon you add a reference to an XML document. You refer to the document by clicking this icon in the bottom of the screen:

Right here:

Please notice: The XML file **MUST** be in the right directory in order to be shown in the designer. The correct directory is /Resources/document in the InterFormNG data directory as specified during installation.

It might be necessary to save the template and close and open the template again to be able to view the XML file in the designer in the **XML Documents** window:

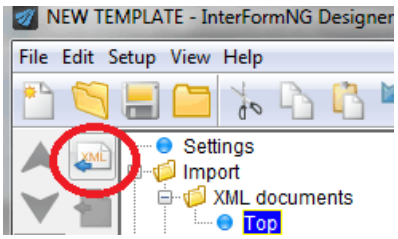


A “Hello World” example

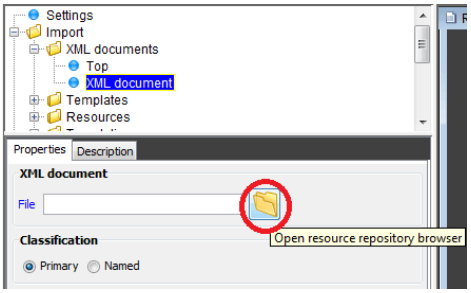
Here we will go through a simple “Hello World” example. We want to create an output with the simple text ‘Hello World’ on a page and then also include a text from an XML file.

Before you start the designer you need to copy some fonts to the /resources/fonts directory as mentioned on page 46. If you have an XML file you can also copy that to the /resources/document directory in the InterFormNG data directory - or you can use one of the demo XML files, which are included in /resources/document/Intro.

Now start the designer. The designer starts up with a new template (=form or overlay). You can now import your XML file by selecting this icon:

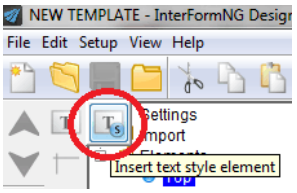


Then select your XML file:

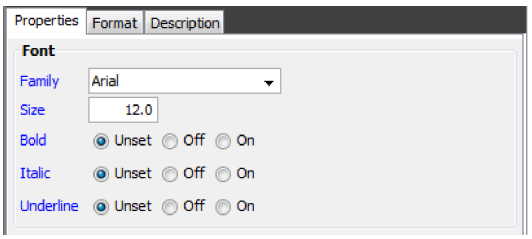


Before proceeding you should now save, close and reopen the template. That will make it possible to see the XML contents in the Designer.

As the next thing you select the text style:

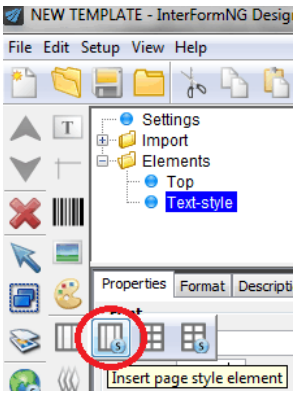


- and specify a font, font size and other attributes you want for the text:



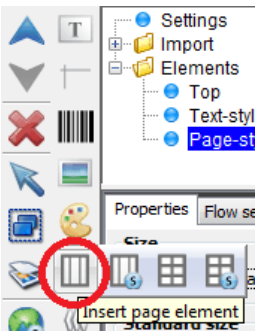
Now you should add a page style to let InterFormNG know how the format/size of the output:

Select this:

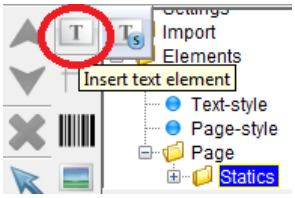


- and specify the paper size to use.

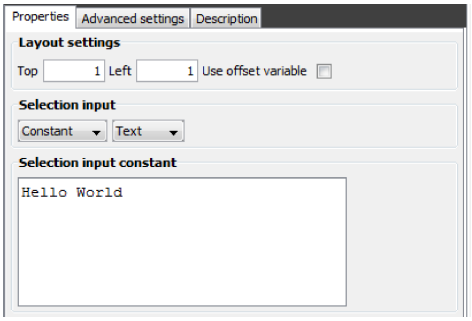
Now add a Page element within which you should insert your design elements:



Then click the ‘Statics’ folder inside the Page element and insert a Text element:

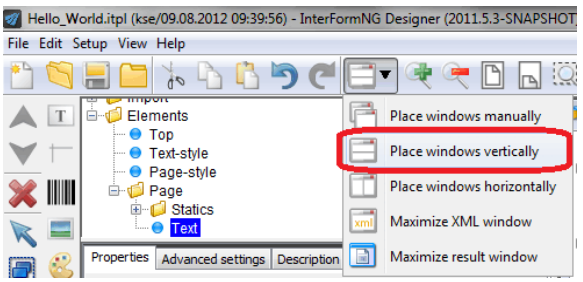


For the Text element you can now simply type a position and the text:



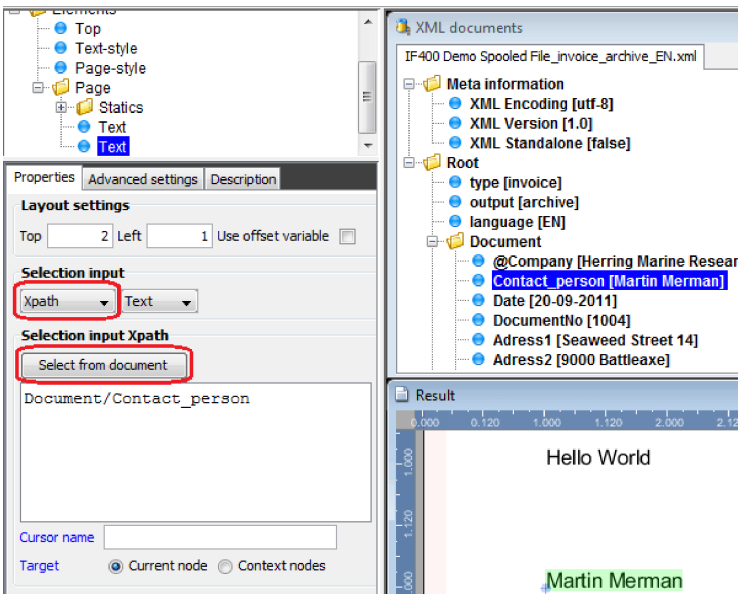
- then click in the result window to tell the designer where to place this text in the output.

If you also want to output data from your XML file you should first display the XML file in the designer. You can e.g. do that by selecting this in the top:



This divides the right of the screen in two: The upper part is the input XML file and the lower part is the result.

If you now select the previous text element in the element list and now insert a new text element, you can now refer to the contents of the XML file:

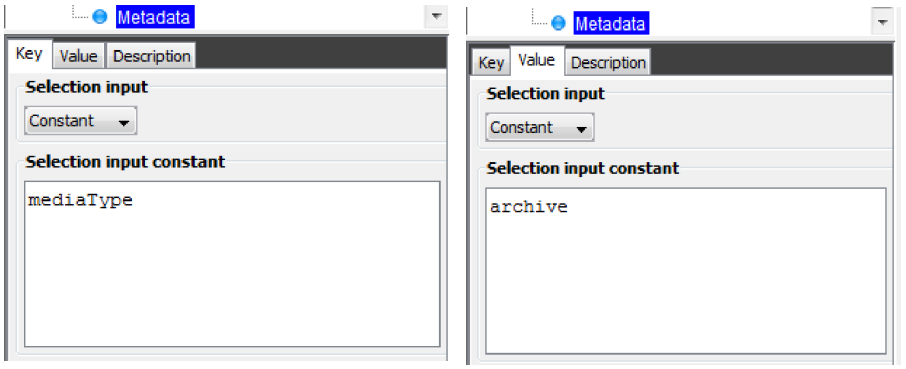


Type the position of this text element and change the selection input into ‘Xpath’. Now you can select contents from the XML file simply by first marking the field in the XML file and clicking the ‘Select from document’ icon and then the path is copied into the Xpath expression. You can also enter this manually or alter this to include various Xpath functions.

Now you can verify the final result in PDF by clicking this icon on the upper right:

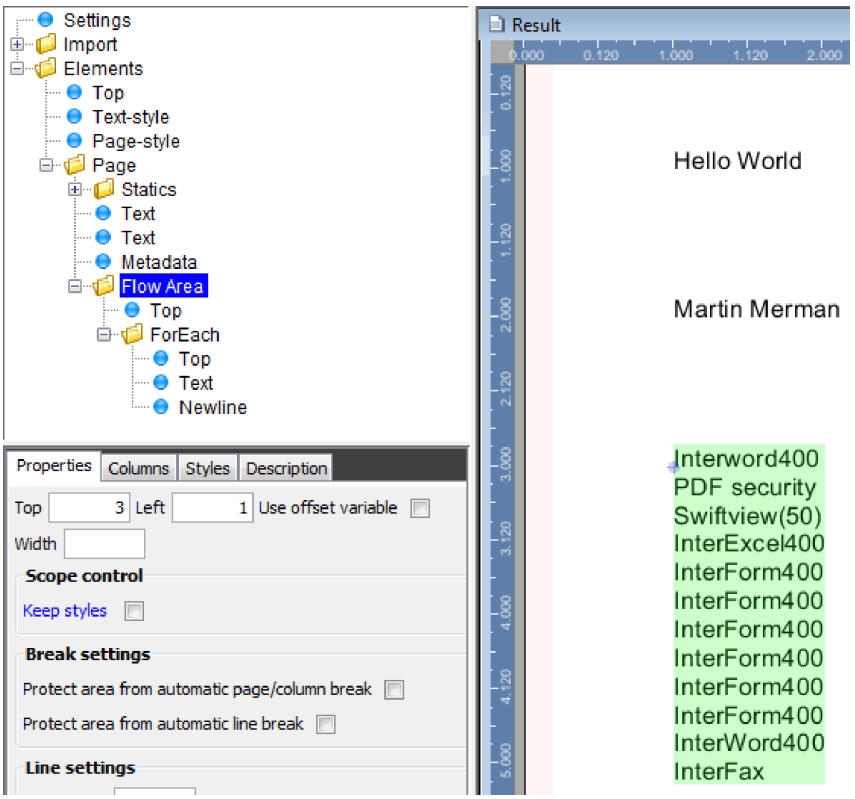
If you get an error regarding a missing print setup you can set the output to be PDF by adding a meta data element. Click on this element on the left:

Now state the meta data key: **mediaType** (case sensitive) and **archive** (case sensitive) as the value as below:



Now you should be able to preview. If not you should verify that the server has been setup correctly and the necessary plugins has been activated.

Now you can continue to build in additional functions your first template. If you e.g. insert a Flow area and inside of this insert a ForEach then you can loop through all nodes with a simular path and print each node with the Text element - remember to insert a NewLine element to force a line break:



Import external resources

When creating your design, you might want to refer to external resources. This can be fonts, templates, images or translation tables. You need to import them first before they can be used in the template design.

Importing a new font

If you want to install a new font to be used in InterFormNG, then you first need to copy the True Type Font file (TTF), onto the font directory inside the configuration directory. The configuration directory is defined during the installation as shown on page 15 and is default: C:\Documents and Settings\All Users\Application Data\InterFormNG

Within the configuration directory you find the font directory here: **\resources\font**, so this is the complete default path:

C:\Documents and Settings\All Users\Application Data\InterFormNG\resources\font

After copying the TTF file here, the font is available for selection in the designer. You just need to restart the designer, if it is already running.

Before you copy any TTF file you should make sure, that you are not violating any copyrights!

Please notice that font styles like italic and bold are only supported as real fonts. Other systems "calculate" italic and bold, and in this case output may look different on different renderers. We have decided to support only real fonts to be sure printers will display output 100% accurate. So you will need to install both the bold and the italic versions of the font, if you want to use them as well.

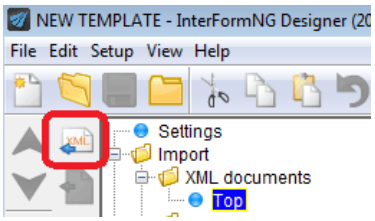
☺ **TIP** ☺

If you only have a TTC file you will need to convert it into TTF. You can e.g. do that with this online converter: http://www.freefontconverter.com/

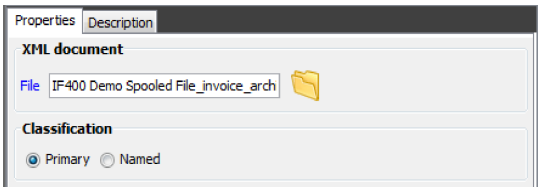
Importing XML file(s)

In order to design your template (form) you will normally need at least one sample XML file, that you want to merge the template with. The XML file(s) must be placed in the /resources/document directory (or a subdirectory inside of this) inside the InterFormNG_Home directory.

You can import an XML file into the template if you open the template and select **Import**, **XML documents** and click this icon:



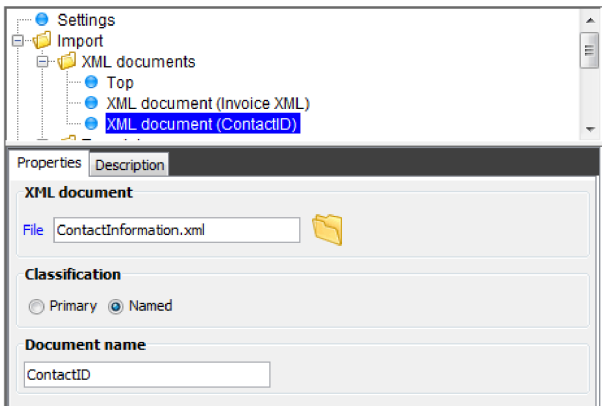
Now you are asked for the XML file, that you want to import:



As default InterFormNG that you will import a primary XML file. If you only want to refer to a single XML file within the template, then it should be a primary XML file. This will be the file, that is substituted when you use the Service to automatically merge incoming XML files with a template.

Using multiple XML files - an example

If you want to be able to refer to more than a single XML file, then the other files should be imported as **Named** XML files. Below an example is used to illustrate the possibilities:

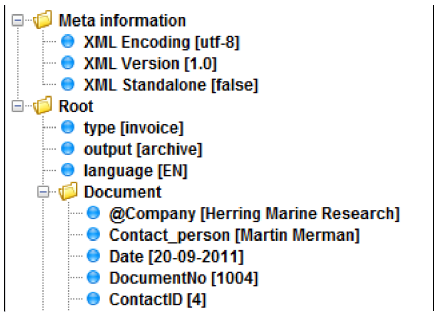


The first XML file (Invoice XML) is the primary file, but in this case we want to refer to a list of the employees of the company to include the name of our contact person in the final output (That information is not found in the invoice - only an employee number).

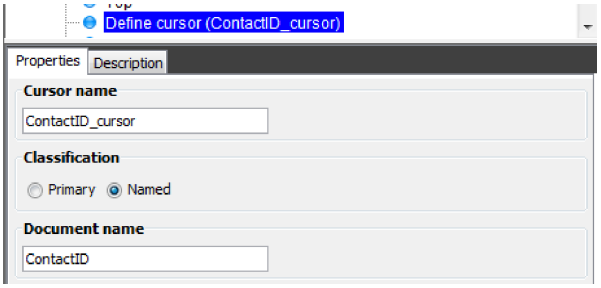
The ContactInformation.xml file looks like this:

```
<?xml version="1.0" encoding="utf-8" ?>
- <Root>
  - <contact>
      <ID>1</ID>
      <Name>Anders Andersen</Name>
    </contact>
  - <contact>
      <ID>2</ID>
      <Name>Børge Brandmand</Name>
    </contact>
  - <contact>
      <ID>3</ID>
      <Name>Christian Christiansen</Name>
    </contact>
  - <contact>
      <ID>4</ID>
      <Name>Daniel Dane</Name>
    </contact>
  - <contact>
      <ID>5</ID>
      <Name>Erik the Great</Name>
    </contact>
  </Root>
```

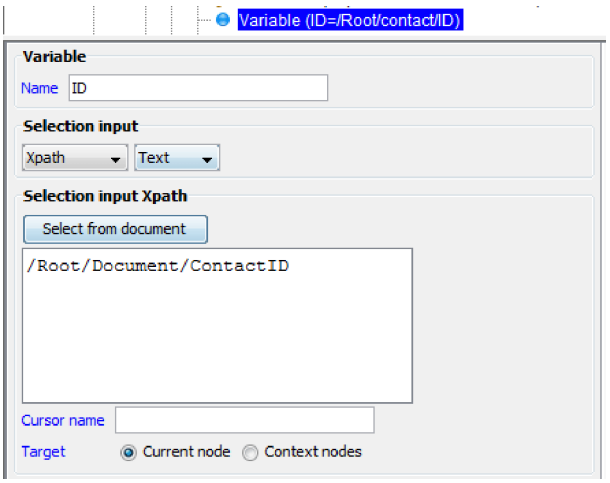
Inside the invoice XML file we have a reference to the ID above - as ContactID (in the bottom):



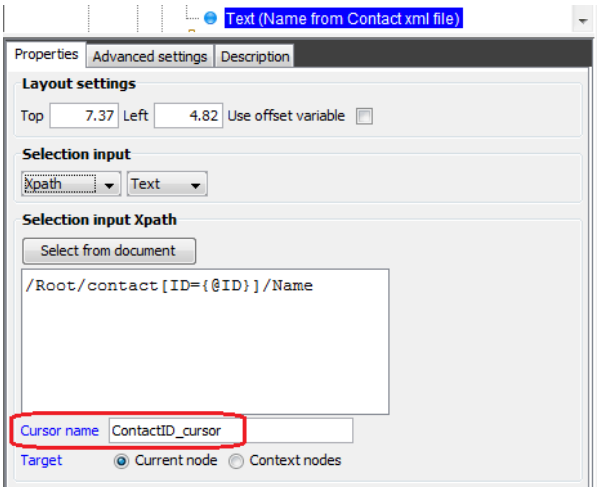
Now we can define an XML cursor to point to the named XML file by clicking this icon:



Next we retrieve the ContactID from the Invoice XML file into variable, ID:



And finally we can do a lookup in the ContactInformation.xml file in this way to print the name of the contact person:



There are three things to notice above:

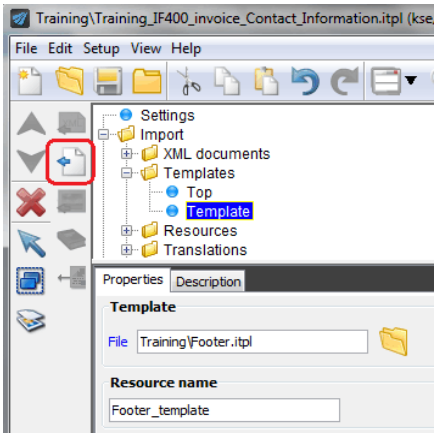
1) The lookup in the ContactInformation.xml file is done by use of variable, ID which is referenced via {@ID} - this is how you refer to a variable in Xpath.
2) The expression /Root/contact[ID={@ID}]/Name returns the value of the Name node, for which the ID is the same as the variable, ID.
3) The cursor name is ContactID_cursor. In this way we are referring to the second, named xml file.

Importing another template

Instead of having the complete design in one single template you can refer to other templates and include them several times in various templates thus reusing the same design many times over. The other template can then in its turn refer to other templates, but you do not need to import them here.

Click on the this element below to import a template:

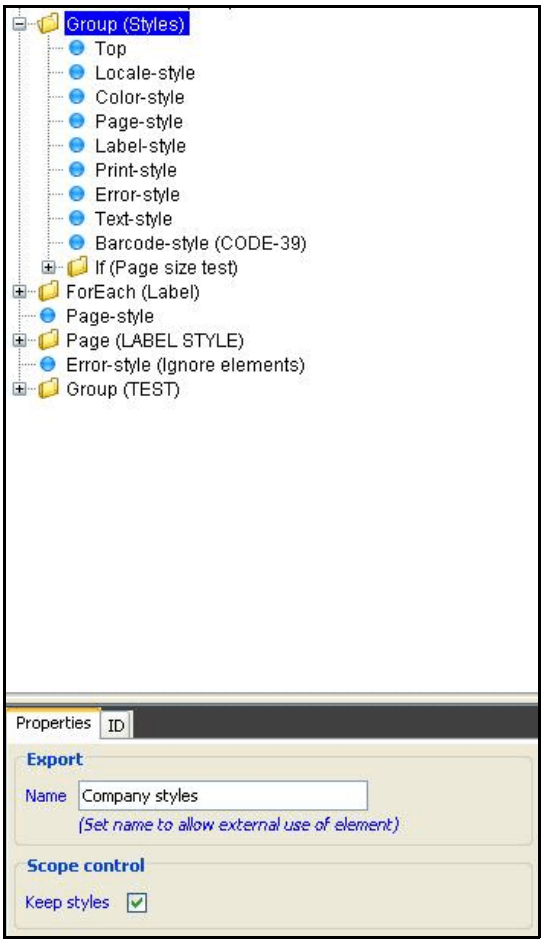
In the properties in the bottom you refer to the template (normally placed in C:\Documents and Settings\All Users\Application Data\InterFormNG\resources\template).



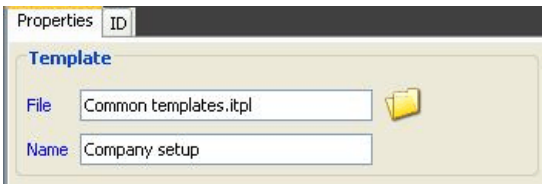
The name you state at the very bottom is the name you refer to in the template when elements from this sub template should be inserted.

You can use the import/export of templates to e.g. define your company styles, which should always referred to or by creating a common header or footer template, that many other templates refer to.

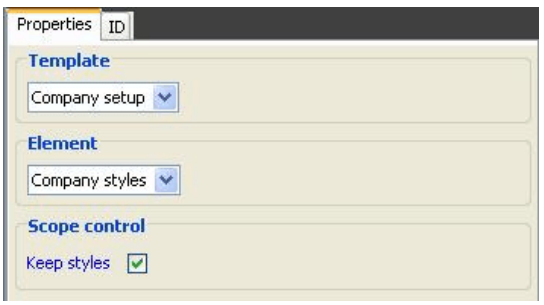
Below an example of such an export is shown:



In the import you can now import these styles to be referred to later via an include element:



The properties of the include element could look like this:



In the example we activate ‘Keep styles’, so that the styles are available outside the included group.

This way of including another template can be used with the main template which you specify for the processor. This template can e.g. inspect the contents of the incoming XML file and include the right template depending on the contents found. (Use the error style to ignore any invalid xpath expressions to avoid an error for other XML files without the same node(s)).

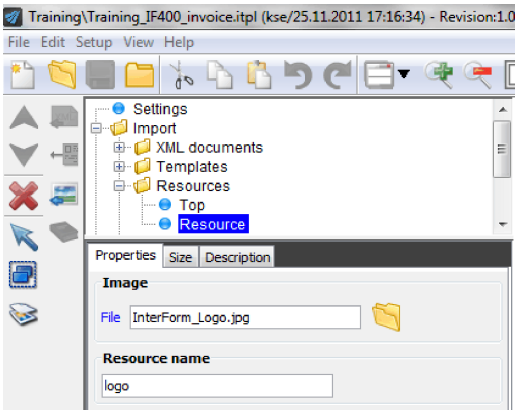
Importing a new image

Images are installed simular to fonts. You copy the BMP/JPG/JPEG/PNG image(s) to the **\resources\image** directory within the home directory (the default path is: C:\Documents and Settings\All Users\Application Data\InterFormNG\resources\image).

WARNING

*Currently **PNG** files are only supported for **PDF output**.*

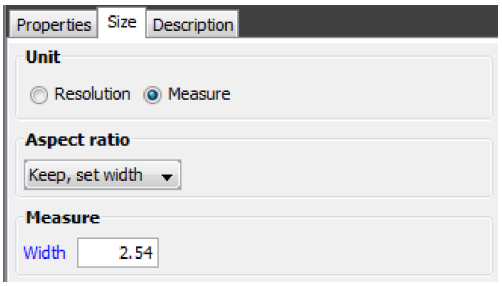
After copying the image you can refer to the image in the designer in this manner: Open the 'Import' tab and the yellow 'Images' folder, click on 'Top' inside and press this icon:



Now the properties window in the bottom is opened and you can open an image file with the open icon. You also state a name for the image, which you can refer to in the template design later.

Setting the image size

You set the size of the image under the 'Size' tab:



Here you can set either a resolution or a fixed size (Measure).

If you select Resolution,then the image size will depend on the resolution selected. The higher resolution you select the smaller the image will be. The reason is, that the image is a certain number of dots in height and width. The resolution is set as a number of dots per inch (DPI), so if you set a large DPI, then each dot will take up a very small space and the image will be small.

When selecting the measure you can set the size of the image via the Aspect ratio.

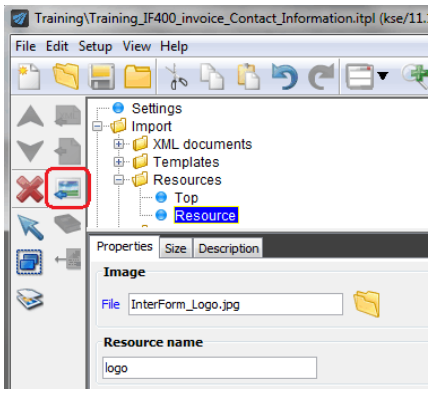
The aspect ratio can be set to these values:
(The measurement used is the measure, that you have chosen under 'Setup' and 'Measure').

Keep, set width	Sets the width below.The height is adjusted to keep the ratio between height and width.
Keep, set height	Sets the height below. The width is adjusted to keep the ratio between height and width.
Do not keep	If you select this you will need to set both the width and the height of the image.

Importing a Resource

If you are going to create PDF files with embedded files, you might want to refer local files instead of dynamically retrieving the files for each preview - e.g. to save time or perhaps these files are not even available in the environment you are currently using.

Such local files must first be imported and this is what you can do by selecting ‘Import’, ‘Resources’, click ‘Top’ and then the icon on the left:



Now you can select any file to be embedded, that are placed in /resources/Design. These imported files can now be referenced for the Link element. Refer to page 82 for details.

Importing a translation table

Translation tables can be used for maintaining texts in various languages. With InterFormNG you can then e.g. let information found in the XML file determine which translation table to use, which e.g. could determine the language of all fixed text elements in the resulting PDF file.

The translation tables are built up of a list of key values and translated values. The **test.properties** file contains this:

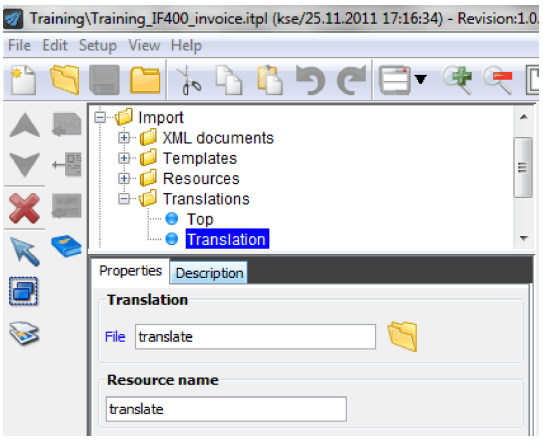
NORWAY = Norway
DENMARK = Denmark
SWEDEN = Sweden
GERMANY = Germany

The leftmost values are the keys and the rightmost are the translated values. The key can be a string without any spaces. The translated value can be a longer text with spaces. Later you can set the locale style in the template and thereby selecting the right translation table.

After you have created a simple text file e.g. in Notepad with keys and values specified like above, you then save it into /resources/translation in the configuration directory as specified during installation. The default full path is:

C:\Documents and Settings\All Users\Application Data\InterFormNG\resources\translation

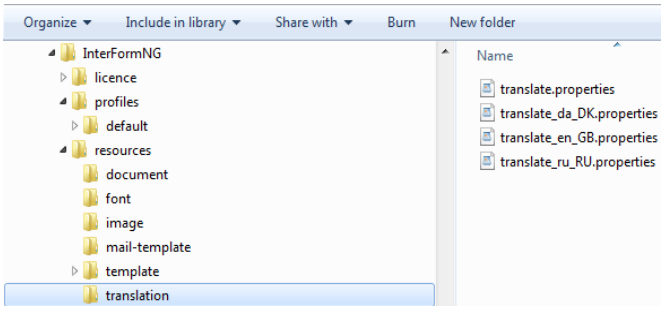
Before you can refer to the translation table in the designer, you need to import it like other resources. So click the 'Import' tab, open the 'Translations' folder, click 'Top' and then this icon on the left:



Now you can refer to the translation file either by typing in the name or by clicking the 'Open' icon and selecting the file. The 'Name' field indicates the internal name to use in the template whenever you want to use this translation table.

Example of translation files

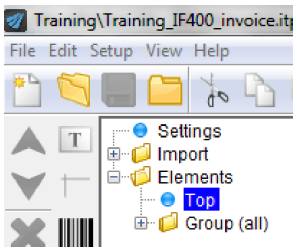
When you refer to the used translation files you refer to the root (or default) translation file. You create an additional file for each language, that you want to handle like below: (Please notice the fixed naming of each file and language below).



As you can see the specific languages need to be included in the file name as shown above. You can use the Locale style to see all possible values for the languages. If you refer to a language, that is not listed, then the default/root file (here: translate.properties) file is used. Also if you refer to a key value not found in the specific language, then the value from the default file is used (if found).

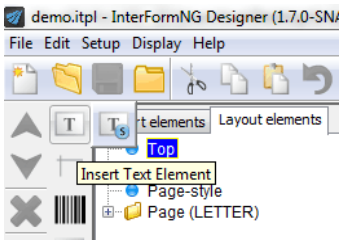
Designing templates

In this section we will describe how you can design the template layouts. In order to start designing you need to open the template (or create a new template) and click the Layout tab in the top then click on 'Top' or any other element below to add or change design elements:



Now the design elements on the left are shown:

If you place the mouse over an icon a pop-up text will describe the icon. If you place the mouse over the rightmost icons additional elements (simular to the first one) is also shown - like this:



In the case above you can see that the current icon is a text element. The icon to the right of it is the text style element (marked with a blue S in the bottom right corner).

The style elements has an effect on the subsequent design elements in the template. This means e.g. that the top text style element with the icon looking like this:

- Can set the font type (e.g. arial), the size and if the texts inserted

now should be bold, italic or underlined. All styles takes effect until another style of the same type overwrites the settings.

It is a good idea to start your template by setting styles for most of the element types - especially the Text style (the icon above) before adding a page element. This makes it easier to find the styles if you later would want to change them. The text style must also be set before the page element to avoid an error.

For the 'physical' elements like e.g. the text element, you should first select the element type by clicking on it with the left mouse key and then click in the result window to state where in output the element is to be inserted. Then the element is added to the element list.

Notice that there is not yet a 'drag and drop' graphical designer, so you need to type in the measurements for positioning by hand. You can however preview the result at any time by clicking the rightmost icon in the upper part of the screen:

- or by looking in the result window as explained below.

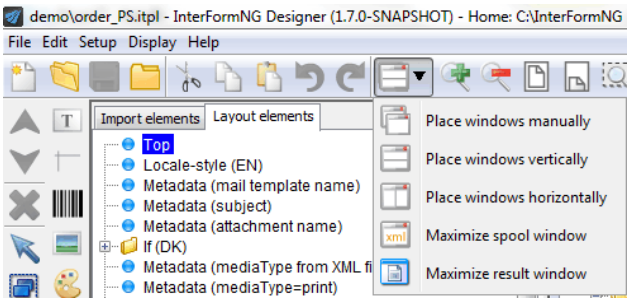
Also notice that all measurements are in inches and pels separated by a comma. 1 inch equals 240 pels, so this means that e.g. 1,120 is the same as one and a half inch. You can change the measurement into either 'normal' inches (without pels) or centimeters as described on page 21.

WARNING

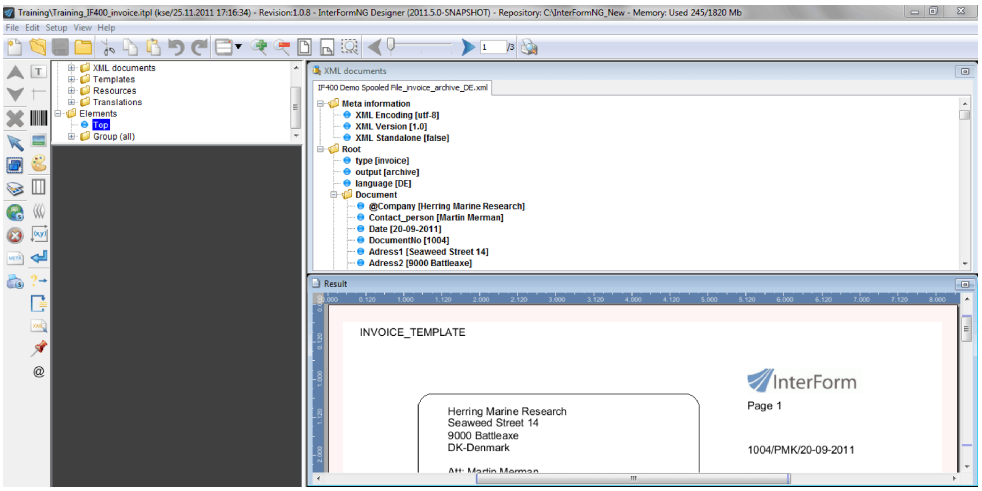
Changing the templates by other means than the InterFormNG designer is unsupported (at your own risk) and may result in unpredictable results.

Viewing Input and/or Result window

Via this drop down menu you can select if you want to view the input XML file and/or the final result in the right part of the screen:



If you e.g. chose 'Place windows vertically' you divide the screen, so you can see both like this:



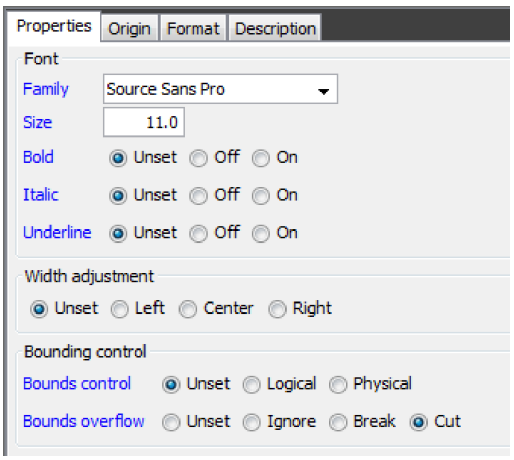
Using Styles in InterFormNG

When you design your template layout, you can use the many styles to tell InterFormNG how to format the output. In this section each style is described.

Any property that is set to the value 'Unset' simply is unchanged, so it will have whatever value it had before this style element, or the default value if it was never set.

Text style

This is a text style. Insert this style to tell InterFormNG what font (and font attributes) you want to use for the subsequent text elements. The style element has these properties:



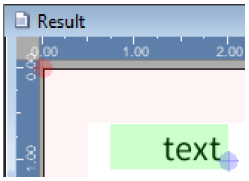
Family
The type of font to use. The possible fonts are limited to the fonts, that you have imported as described on page 46. You can also specify 'Unset' for this field, if you want to keep a font family previously set.

Size
The size of the font measured by the height in dots.

Bold/Italic/Underline
Chose if the font should be Bold, italic or underlined.

Width adjustment
If you have set the text element to a fixed width like described on page 73, then you can use this to horizontally position the text inside the reserved area.
Left: Positions the text on the left.(**Default**)
Center: Centers the text relative to the area.
Right: Place the text to the right inside the area.

Example: This is a text with a fixed width of 120 pels (½ inch), which is using width adjustment=Right:



The green box shows the area for the text element. The blue dot shows, where the text is positioned (1 cm from the top and 2 cm from the left). The horizontal adjustment mentioned below will set the origin (alignment) of the complete area.

Bounding control

Bounds control

The bounds control sets how the boundaries (or actually the preceeding/trailing spaces) of the text should be considered. The values are:

Logical (Default): The trailing/leading spaces are a part of the text and will be used when aligning.

Physical: Any leading/trailing blanks are ignored when you set the text to be left or right aligned. (Currently the leading blanks are not ignored..)

You can at the moment think of this parameter as 'Ignore trailing blanks', where 'Physical' sets this to 'Yes'.

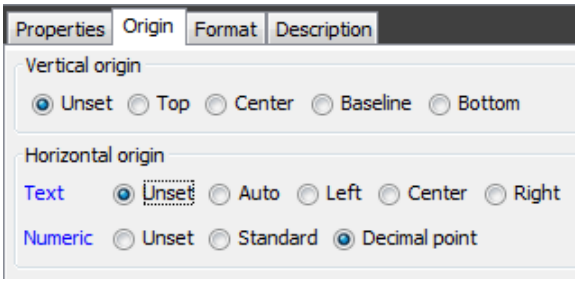
Bounds overflow

On the text element you can (under advanced settings) set a maximum width of the text element. If you do that, then this setting is used.

Break:If you set it to 'break', then a line break (like the NewLine element) will be added when the max. width is reached. You should only use the 'break' option within a flow area to make sure, that the line break will do a line feed.

Cut: Set it to '**cut**' to simply cut the text element at the specified width.

On the Origin tab of the text style you see this:

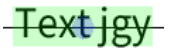


Vertical origin

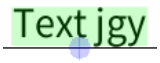
What reference point to use when positioning the text.

Top means that the top of the text will be placed on the vertical position stated and that the text will 'grow' downwards if you later increase the size.

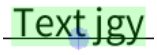
Center The middle of the text is used.



Bottom The very bottom of the text is the reference/origin.



Baseline The baseline of the text is used as reference - as shown below:

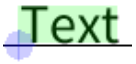


You should avoid using 'Bottom' unless necessary as this is not supported for rotation 90,180 and 270 for ZPL output.

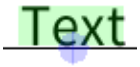
Horizontal origin
How to place the text relative to the position stated. Here you can e.g. make the text right adjusted or centred.

Auto Works like left origin.

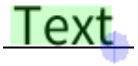
Left (Default). Use the left of the text as origin/reference.



Center Centers the text around the origin.



Right Use the right of the text as the origin:



Numeric origin
The numeric origin setting is used for texts, that have been set to be numeric. It sets the horizontal alignment for those
The possible values are:

Standard (Default) Use whatever is setup for normal texts, which are as default using the left edge as the origin.

Numeric Use the decimal point, which must be a dot (.) for the texts, that has been defined as numeric.

Format
The Format folder contains possible ways to represent dates, times and numbers. In order to format a date, time or number, then the data must fit the xml requirements for a valid date, time or number. You can chose to specify a mask for the output.

Numeric format
To recognize a number as numeric the format of the data must be -###.## i.e. no thousand separator and a dot as decimal point and a preceding minus for negative numbers. However if you want a different format in your output you can use the table below for a reference of how you can alter various input formats into a new output format:

		Input format			
		1234.5 (numeric)	1,234.5	1.234,5	1234,5
Output format	1234.5	Mask: ####.# Locale: en-US	translate("{@in}",",","")	translate ("{@in}",",",".")	translate ("{@in}",",",".")
	1,234.5	Mask: #,###.# Locale: en-US	No action needed	translate ("{@in}",",",".,")	translate ("{@in}",",",".") (*)
	1.234,5	Mask: #,###.# Locale: de-DE	translate("{@in}",",.",".,")	No action needed	translate ("{@in}",",",".") (*)
	1234,5	Mask: ####.# Locale: de-DE	translate("{@in}",".,",",")	translate ("{@in}",".","")	No action needed
	1234.50	Mask: ####.#0 Locale: en-US	translate("{@in}",",","") Mask: ####.#0, en-US	translate ("{@in}",",",".") (*)	translate ("{@in}",",",".") (*)
	Numeric calculations	No action needed	translate("{@in}",",","")	translate ("{@in}",",",".")	translate ("{@in}",",",".")

These tricks can be used for setting the output format:
1) In the formulas above **{@in}** indicates a variable, **in** that contains the input data.
2) The function, translate is able to translate one character into another. This will

e.g. replace any comma with a dot in the variable, in:
`translate("{@in}", "," , ".")`
Refer to page 124 for details.
3) The locale style decides what characters the decimal point and the thousand separator should be (if used). Above en-US is used for a comma as thousand separator and a dot as decimal point. The de-DE selects the opposite. You may use your own locale - if it fits the output you want.
4) The trailing 0 in the format ###.#0 selects a 2 decimal format. This is e.g. often used for amounts.
5) Negative values must be written with a preceding minus to be recognized correctly. If you want a different output format you write the format for negative values after a semicolon like e.g.: ###.#;###.#- (if you want a trailing minus for negative values).
6) For all cells marked with a (*) above you need to use the same numeric formatting as in the leftmost column.
7) To use the numeric formatting the text element must also be set as numeric.
8) For most of the formulas above the result is not 'real' numeric i.e. it cannot use the numeric formatting. If you want to use numeric formatting use the formulas in the bottom row (numeric calculations) and use the numeric mask in the leftmost column.

Date format
A date must be written in the format YYYY-MM-DD (Year-Month-Day) with preceding zeroes i.e. 2014-08-31 is valid while 2014-8-31 is not valid. The input date format found in the XML file might however be different from this and the requested output format might of course also be different. Refer e.g. to this link for explanation of how to type a date mask:
http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html

The table below should also help you to convert dates. You can convert the date format in two ways:
1) Use the concat and substring xpath functions to build up the requested format - this is not always simple.
2) Convert the input format into the correct format (YYYY-MM-DD) and convert the date into the right output format via the formatting in the text style.

		Input format				
		2014-08-31 (date format)	2014-31-08	08-31-2014	31-08-14	31-8-14
Output format	2014-08-31	No action needed	concat(substring("{@date}",1,4),"-",substring("{@date}",9,2),"-",substring("{@date}",6,2)	concat(substring("{@date}",7,4),"-",substring("{@date}",1,2),"-",substring("{@date}",4,2))	concat("20",substring("{@date}",7,2),"-",substring("{@date}",4,2),"-",substring("{@date}",1,2))	concat("20",substring-after(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before(substring-after("{@date}","-"),"-"))),substring-before(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before("{@date}","-"))),substring-before("{@date}","-"))
	2014-31-08	Mask: yyyy-dd-MM	No action needed	concat(substring("{@date}",7,4),"-",substring("{@date}",4,2),"-",substring("{@date}",1,2))	concat("20",substring("{@date}",7,2),"-",substring("{@date}",1,2),"-",substring("{@date}",4,2))	concat("20",substring-after(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before("{@date}","-"))),substring-before("{@date}","-"),"-",substring("0",1,2-string-length(substring-before(substring-after("{@date}","-"),"-"))),substring-before(substring-after("{@date}","-"),"-"))
	08-31-2014	Mask: MM-dd-yyyy	concat(substring("{@date}",9,2),"-",substring("{@date}",6,2),"-",substring("{@date}",1,4))	No action needed	concat(substring("{@date}",4,2),"-",substring("{@date}",1,2),"-","20",substring("{@date}",7,2))	concat(substring("0",1,2-string-length(substring-before(substring-after("{@date}","-"),"-"))),substring-before(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before("{@date}","-"))),substring-before("{@date}","-"),"-","20",substring-after(substring-after("{@date}","-"),"-"))
	31-08-2014	Mask: dd-MM-yyyy	concat(substring("{@date}",6,2),"-",substring("{@date}",9,2),"-",substring("{@date}",1,4))	concat(substring("{@date}",4,2),"-",substring("{@date}",1,2),"-",substring("{@date}",7,4))	concat(substring("{@date}",1,2),"-",substring("{@date}",4,2),"-","20",substring("{@date}",7,2))	concat(substring("0",1,2-string-length(substring-before("{@date}","-"))),substring-before("{@date}","-"),"-",substring("0",1,2-string-length(substring-before(substring-after("{@date}","-"),"-"))),substring-before(substring-after("{@date}","-"),"-"),"-","20",substring-after(substring-after("{@date}","-"),"-"))
	14-08-31	Mask: yy-MM-dd	concat(substring("{@date}",3,2),"-",substring("{@date}",9,2),"-",substring("{@date}",6,2))	concat(substring("{@date}",9,2),"-",substring("{@date}",1,2),"-",substring("{@date}",4,2))		concat(substring-after(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before(substring-after("{@date}","-"),"-"))),substring-before(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before("{@date}","-"))),substring-before("{@date}","-"))
	31-08-14	Mask: dd-MM-yy	concat(substring("{@date}",6,2),"-",substring("{@date}",9,2),"-",substring("{@date}",3,2))	concat(substring("{@date}",4,2),"-",substring("{@date}",1,2),"-",substring("{@date}",9,2))		concat(substring("0",1,2-string-length(substring-before("{@date}","-"))),substring-before("{@date}","-"),"-",substring("0",1,2-string-length(substring-before(substring-after("{@date}","-"),"-"))),substring-before(substring-after("{@date}","-"),"-"),"-",substring-after(substring-after("{@date}","-"),"-"))

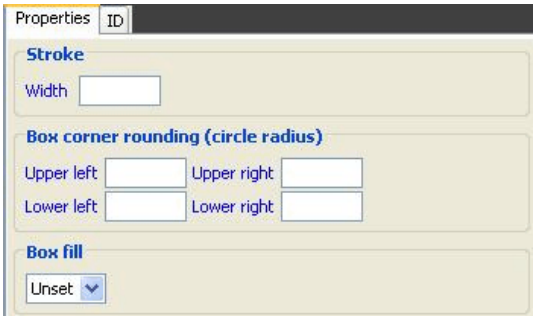
		Input format				
		2014-08-31 (date format)	2014-31-08	08-31-2014	31-08-14	31-8-14
Output format	31-8-14	Mask: d-M-yy	Use date format	Use date format	Use date format	No action needed
	Sunday, 31. August 2014	Mask: EEEE, d.MMMM YYYY	Use date format	Use date format	Use date format	Use date format
	Sun., 31. Aug. 2014	Mask: EEE., d.MMM YYYY	Use date format	Use date format	Use date format	Use date format
	August 31. 2014	Mask: MMMM d. YYYY	Use date format	Use date format	Use date format	Use date format
	Date format	No action needed	concat(substring("{@date}",1,4),"-",substring("{@date}",9,2),"-",substring("{@date}",6,2))	concat(substring("{@date}",7,4),"-",substring("{@date}",1,2),"-",substring("{@date}",4,2))	concat("20",substring("{@date}",7,2),"-",substring("{@date}",4,2),"-",substring("{@date}",1,2))	concat("20",substring-after(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before(substring-after("{@date}","-"),"-"))),substring-before(substring-after("{@date}","-"),"-"),"-",substring("0",1,2-string-length(substring-before("{@date}","-"))),substring-before("{@date}","-"))

For the dates these tricks are used:

1) For the formulas above **{@date}** indicates a variable, **date** containing the date. You should replace this with the XML node or your variable.
2) The concat function concatenates (puts together) multiple string into one.
3) The substring(string, start, length) function returns a part of the string, starting at position ‘start’ and with the length of ‘length’.
4) The function, substring-before(string,search-string) searches for the search-string in ‘string’ and returns the part of string, that is to the left of the search-string.
5) The function, substring-before(string,search-string) searches for the search-string in ‘string’ and returns the part of string, that is to the left of the search-string.
6) string-length(string) returns the length of a string.
7) For the cells above marked with ‘Use date format’ you can get to the output format by selecting the date format function and use the date format ask in the leftmost cell in the same row.

Geometry style

The geometry style sets attributes of subsequent lines and boxes.
The properties are these:



Stroke
The width of the lines used. This is measured in whatever measurement you have chosen, so usually this will not be a whole number but rather e.g. 0.05 or simular.

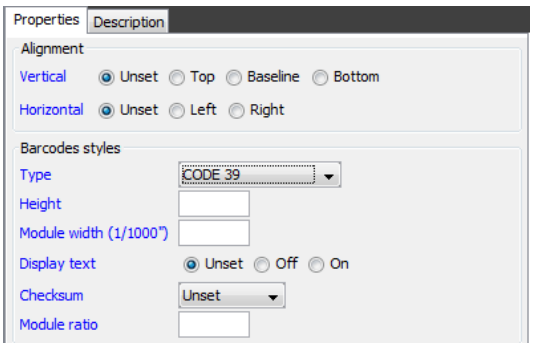
Box corner rounding (circle radius)
The radius of the circle for which a circle arc is used as a corner. Specify 0 for a sharp corner or the radius of the circle for a round corner.

Notice: Round corners are not supported for ZPL output. Any rounded corners are in ZPL printed as sharp.

Box fill
Set this to 'On' if you want to fill out the inside of the boxes inserted after the style.

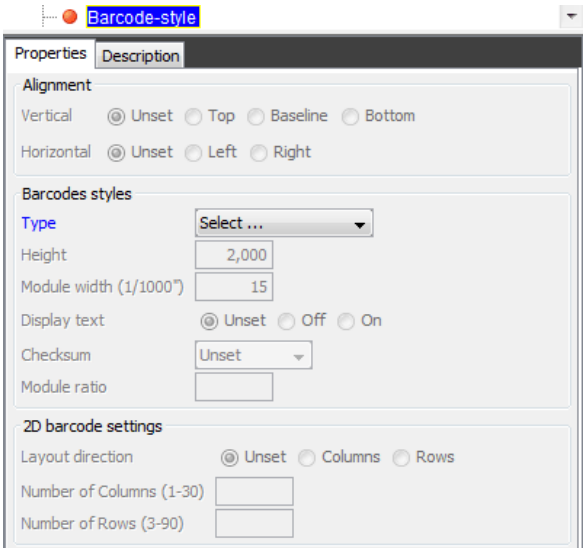
Barcode style

The barcode style determines what kind of barcode you insert the coming times a barcode element is executed. The properties of the barcode style are these:



Alignment
How the barcode will be positioned relative to reference point. You should avoid using 'Bottom' unless necessary as this is not supported for rotation 90,180 and 270 for ZPL output.

Type
The type of barcode e.g. Code39, Code128 etc.. The displayed selection of the remaining options will depend on what you select here. If you have upgraded to a 2014 version (or newer) from a pre 2014 version and you defined the type to be 'Unset', then the barcode style will look like this:



As you can see the 'Unset' value is shown as 'Select...' and the element is shown in red to indicate an error. This barcode style is ignored until you select a specific barcode type. After selecting a specific type you can also edit it.
For details of how barcode styles are converted please refer to page **165** in **Appendix H**.

Height
The height of the barcode in the measurement you have chosen via 'Setup' and 'Measure'.

Module width
The size of the smallest bar/unit in the barcode. This is measured in 1/1000 parts of an inch, so that a width of 1000 equals 1 inch. InterFormNG will select a width that is as close as possible to this value - rounded up to the nearest number of whole dots. The width of a dot depends on the printer resolution (and for PDF the printer style setting). Here are the widths of a dot in some resolutions:
1 dot in 203 dpi is (1/203) = 0.0049 inch wide
1 dot in 300 dpi is (1/300) = 0.0033 inch wide
1 dot in 600 dpi is (1/600) = 0.0017 inch wide

So the width of a barcode may vary depending on the select module width and the printer resolution setup in output configuration (in the web interface).
Some 'golden' module widths should be considered if you want the barcode size to be as close as possible for 203,300 and 600 dpi. These 'golden' values are: 4, 9,12,13,19,22,23.

If you want to see a barcode size close to the actual output it is a very good idea to select the same resolution in a print style (refer to page **69** for details).

Display text
Activate this if you want the text in the barcode to be written as human readable text below the barcode.

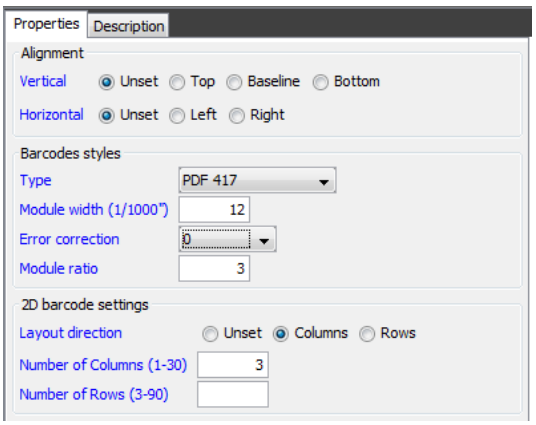
Checksum
With this you can chose to add a checksum to the barcode. The possible values are:
Unset: Keep the previous value. As default a checksum will not be added.
Check: Tests the checksum of the input data. Invalid checksum results in an error.
Ignore: The checksum is not tested and is not added unless it is included in the input data. This is the default.
In pre 2014 versions you could also select '**Auto**' as the value. This is no longer supported, so if used the barcode style will be marked in red and an error will

occour after an upgrade: **001010:Invalid barcode (Illegal value for checksum: AUTO)** will be issued - until you change the value.

Module ratio
The module ratio is the ratio between the wide and narrow lines in the barcode. A value of e.g. 2,5 will make the wide lines 2,5 times as wide as the narrow lines. The ratio will be rounded down to fit a whole number of dots in the specific printer resolution (as specified in the output configuration).

For the preview and result wind the size of the barcodes are mainly defined by the settings in the barcode style, but is also affected by the printer resolution set in the print style (refer to page 69), so it is recommended to set the print style to the actual resolution you want to use for printing.

For the 2D barcodes (PDF417 and QR) a few extra fields may appear on the barcode style:

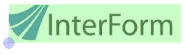


Error correction
This sets the error correction for the barcode. The Error correction for a PDF417 barcode can be set as a value between 0 and 8. The larger the value the larger the barcode will be - and easier to scan. The Error correction for QR can be L, M, Q or H (listed after size).

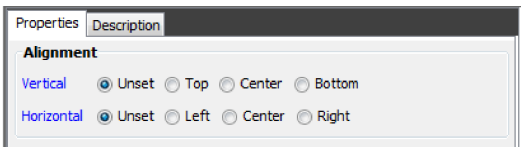
Layout direction
The layout direction can be set for the PDF417 barcode. This sets in which direction the barcode will 'grow' - or rather a maximum height or width.
Columns (default) sets a fixed (or rather maximum) width.
Rows sets a fixed (or rather maximum) height.
The selected layout direction decides if the specified **number of columns or rows** will be used (the other will be ignored).

Image Style

As default the reference point of an image is the lower left corner:



With the Image style you can change the reference point:



This has effect on the next image elements inserted. Refer to page 77 for details of the image element.

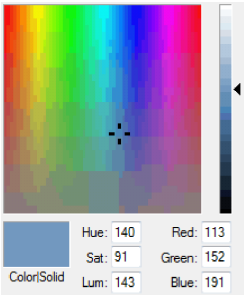
Color style

Set the color of the following elements and the color to be used when filling out boxes:

Instead of selecting one of values from the drop down list you can also enter the 2 digit hexadecimal values for the Red, Green and Blue combination of the color you want. Here are a few examples:

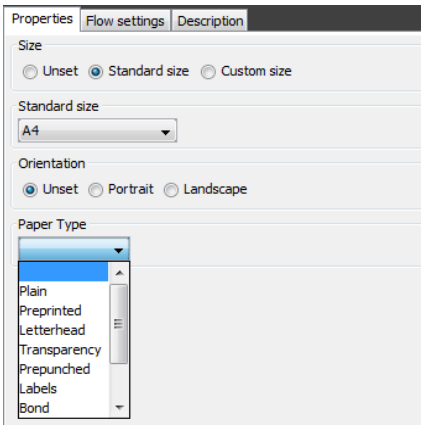
Red: **FF0000**
Green: **00FF00**
Blue: **0000FF**

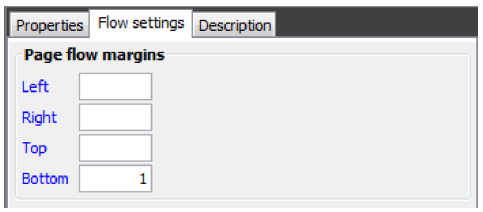
The color on the right:
7198BF



Page Style

The page style sets the page size (e.g. A4, Letter etc...) and orientation (portrait or landscape). You can also select Custom size to select a specific output size (you are then asked for the height and width of the output). As with most styles both can also be set to 'Unset' indicating that this property should not be changed and the previous set value is to be used.



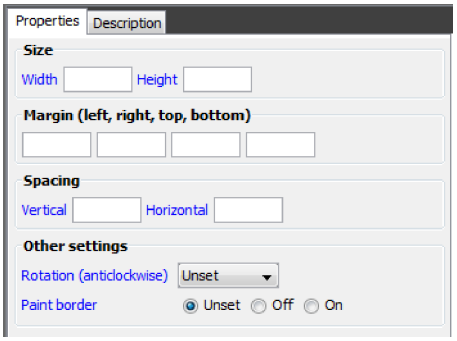


Paper Type: This selects the paper type to use when printing in PCL. You can use this to **select the drawer (or tray)** from which the paper should be taken. You need to select the equivalent paper type on the printer you intend to print out on. The paper type is language independent, so you need to select the similar type of paper on the printer menu in your own language if the printer menu is not in english.

Under **Flow settings** you can set the bottom margin for any flow area within the page element. This margin will be added to the bottom padding of the flow area.

Label Style

The label style sets the way you want to place labels on the resulting page. You can decide to put multiple labels on one page. The properties are these:



Width
The width of one label. State this in inches as pels - as all the other measurements.

Height
The height of the label.

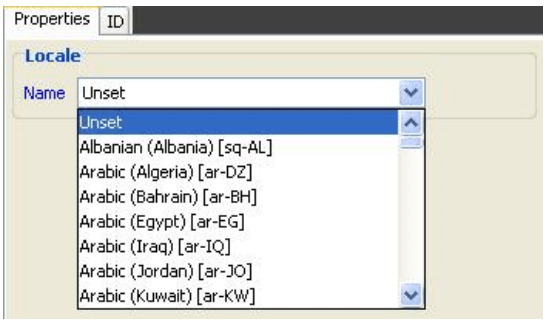
Margin
The margins for the page. This is the empty space that is reserved around the edge of the output page - not be used for labels.

Spacing
The free space between the labels - measured also in inches and pels.

Border
You can chose to print lines at the border of each label.

Locale Style

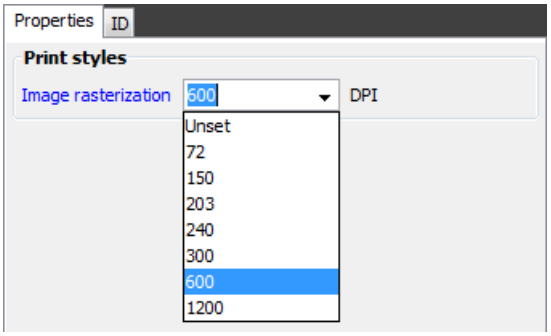
The Locale Style sets the location which can be used for triggering the right translation table. It can e.g. be combined with a conditioned section that determines the location from contents in the XML file. The property of this style is simply the location as selected via the drop down list as shown below:



The locale style also affects the output of numbers and dates, so that they are outputted in the format used locally.

Print Style

The Print Style sets the resolution to be used for any included images. This also sets the resolution of the barcode images that InterFormNG is generating for the PDF output. In order to get the right barcode output, it is very important to select the right resolution to avoid resizing. Resizing can result in possible barcode scanning problems.

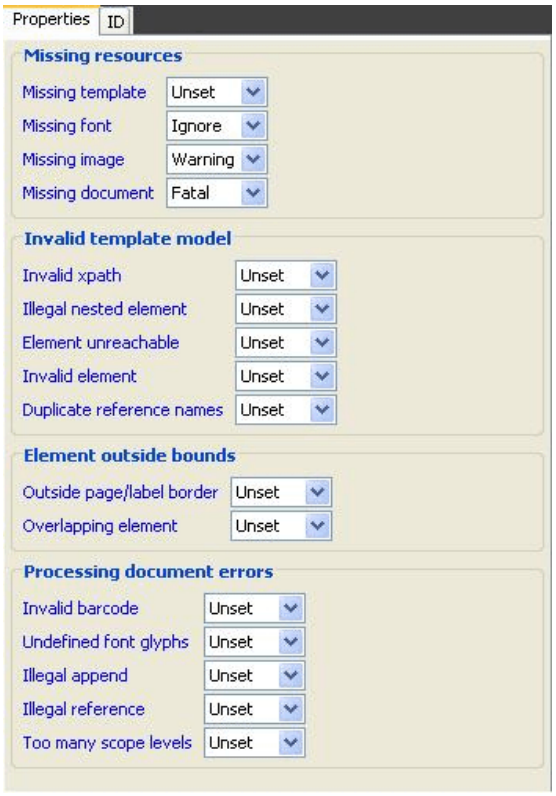


The default value (if no print style is specied) is 600 dpi. It is recommended to select a value, which is dividable with the printer resolution without remainder, so for a 600 dpi result you should avoid 72, 203, 240 and 1200 dpi - especially if you are generating barcodes. If you select one of these values for the PCL output, then the barcode will be scaled to fit the output resolution, which will create a barcode of a different size compared to PDF and Postscript (to make the barcode as scannable as possible).

Reduce the output size
If the output is getting too large or if you have performance issues, you can reduce the resolution and check if the result is still acceptable - remember to test all types of output that you want to generate (PCL, PDF and Postscript).

Error Style

The Error Style tells InterFormNG what to do if a problem should occur. The list of monitored errors can be seen here:



For each possible problem 4 settings are possible:

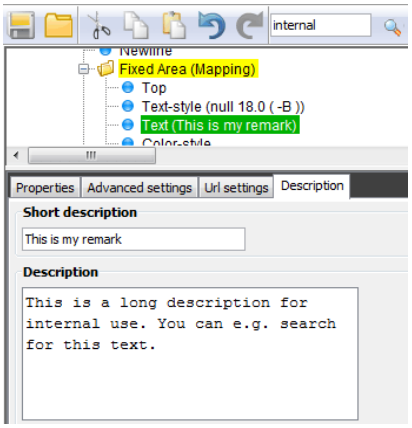
Unset This is undecided. Use whatever value that may have been selected earlier.

Ignore Ignore the problem and proceed.

Warning Issue a warning in the log, but continue processing the file.

Fatal Stop processing and issue a fatal error.

For all elements (and styles) you can state a **short description**:



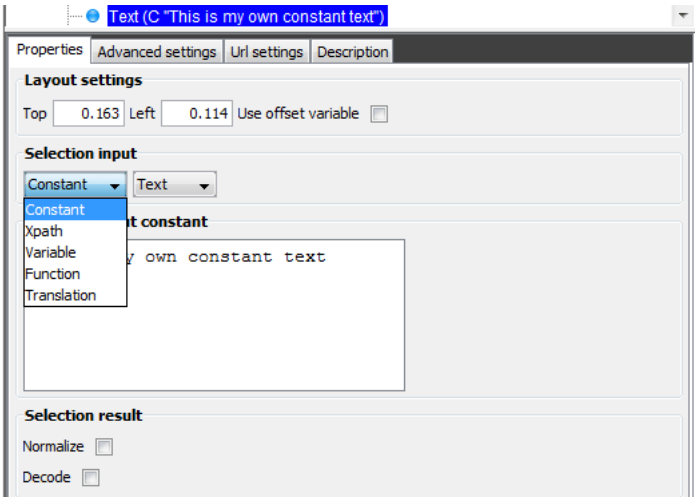
- By clicking the Description tab. This will be shown in the list of the elements like shown above. If you do not add a short description the properties of some elements (texts and text styles) will be shown in the list instead.

You can also add a longer description which will not be shown (unless you click the Description tab). You can search for this and other settings and contents via the **search field** in the top of the screen.

The design elements of InterFormNG (which can be found on the left in the designer) are listed below from the top and down. Most of them are inserted by first clicking the element type on the left and then the position in the result window.

Text Element

The Text element simply inserts a text in the output. The font used will be the one specified in the last text style element. The origin of the text is set in the properties of the element. It can be one of the below:

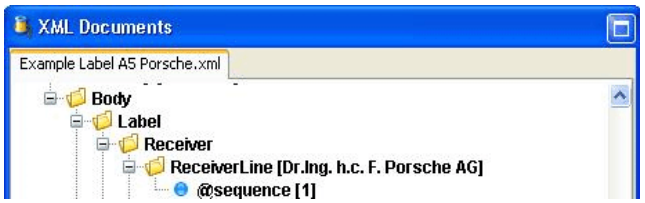


A constant text. Simply type the text to be inserted.

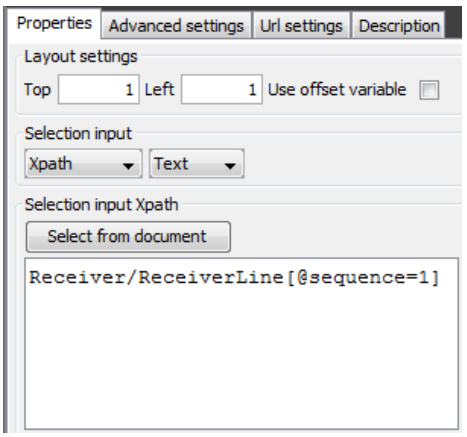
Xpath. A text/value from the XML file is inserted in the output. If you select the Xpath option, you will be asked to specify the XML value via Xpath (the path in the XML file) and a cursor reference name. The Xpath defines the path from the referenced cursor - or if this field is blank: The path from the default cursor. The cursor is a 'pointer' pointing to the place in the XML file, that is currently worked on.

If you want to refer to the value of a variable in an Xpath expression, you state it like this: {@variable}. This will be substituted before the Xpath expression is calculated.

Below the cursor is set via a previous 'For Each' element, which has set the cursor to the 'Label' branch. This is the XML tree:



Below the text 'Dr.Ing.h.c. F. Porsche AG' in the output by referring to the xml contents.



The Xpath expression refers to Receiver/ReceiverLine inside the 'Label' node. The [@sequence=1] tells InterFormNG to only select the ReceiverLine node(s), where the sequence is 1.

The simplest way to refer to xml nodes is to click on the node in the XML tree and then click on the icon **Select from document** in the text element above. Refer to **Appendix A** on page 123 for additional information about Xpath.

Variable. Include the value of a variable in the output. If you want to combine the value of the variable you should chose Xpath instead.

Function. This refers to an internal function in InterFormNG. Available functions are: PAGE_NUMBER, PAGE_NUMBER_TOTAL and TIMESTAMP. The function, PAGE_NUMBER will return the current page number of the output making it possible to e.g. print the number via the text element. The function PAGE_NUMBER_TOTAL inserts the total number of pages in the output. This will show up as 99999 in the result view in the designer and only show the correct in the final result. The TIMESTAMP function returns the current date and time. You select the function via a drop down list.

Translation. You can refer to a translation table. If you select this you are asked for a resource and a key. The resource field is one of the available translation tables. In order to make a translation table available you need to import it first. The key is the key value for which you are doing a lookup in the translation table. The resulting value is inserted in the output. The translations are described on page 54.

Data type
To the right of the element type (Constant/Variable, Function..), which is described above, you set the data type of this element. This can be set to Text, Boolean, Number or Date. The data type will have an effect on how the data is

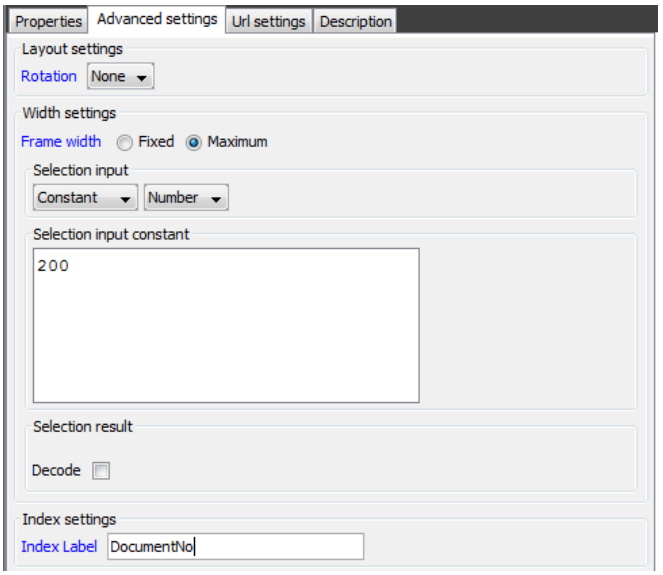
printed out, but you can also use numeric Xpath functions if you set the data type to numeric.

It is very important, that the text really is the correct type when defining it as boolean, number or date. The requirements are these:

Boolean	Values **0** and **1** or a boolean xpath expression are allowed. Invalid values are currently not shown and no error message is issued.
Number	Numeric values using a **dot as a decimal point** - without any thousand separator are allowed. Invalid data are currently not shown and no error message is issued.
Date	Only dates in the format **YYYY-MM-DD** is supported. Invalid dates/strings may result in a strange date to be shown.

In the text style (under 'format') you can then even tell InterFormNG how to present this specific data type. Refer to page 58 for details.

Advanced settings



The options are placed under the **advanced settings tab** of the text element:

Rotation
The rotation of the text element. Possible rotations are: None, 90, 180 and 270 degrees.

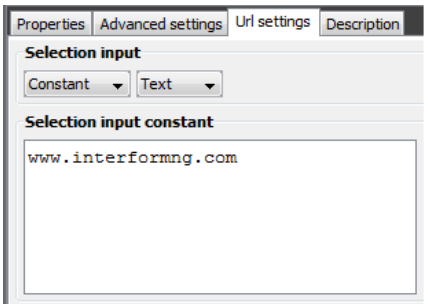
Frame width
This is the width of text. If you leave it blank, then the width of the text is undefined i.e. it will be as long as needed.
With a value below you can chose to set either a **fixed or a maximum** width:
Fixed width: If you set the width value to be a fixed width, then the width of the text will be exactly as wide as the size set below. If you use the text inside a flow area, then you can use a fixed width to set columns for some detail lines - if you set the bounds overflow (on the text style) to either 'cut' or 'break'. If you want to align the text inside the designated 'frame', then you need to use the **Width adjustment** option of the text style.
Maximum width: The maximum width allowed of the text element. This option also works closely with the **Bound overflow** option of the text style. Refer to the text style on page 58 for details.
Selection input: In the bottom you set the width of the text counted as 'pels'. (240 pels equals 1 inch, which is 2,54 cm.). Remember to set this to be a **number**. The size above of 200 pels equals 0.83 inches or 2.12 cm =2.54*200/240.

You need to **remember to set the type to number** (instead of the default 'text'). If you forget, then this width setting is ignored.

Index Settings - Index label
In the bottom of the advanced tab (of the text element) you can decide if you want to include the text element in an xml file, that is intended for **archiving in e.g. in InterArchive**. The output xml file is created if the index label is defined for one or more text elements and only if you are creating an output PDF file.

The output XML file will be created in the same directory as the output PDF file and with the same name - except the extension .xml. The XML file contains each index label and the value (the text that is the output in the text element) and some general meta values e.g. the name of the original input xml file, the path of that input file and the total number of pages.

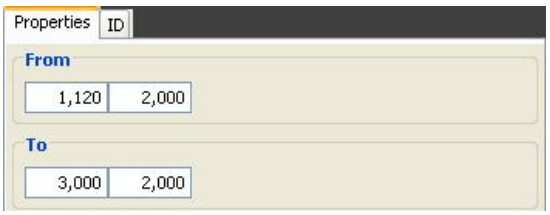
Url Settings



You can add a link in the PDF output for the text via the Url settings tab above. The actual link can be a constant like above, an Xpath expression, a variable, function or translation. Please note, that the url will not be highlighted as default (e.g. in blue with underscore), so if you want that you will need to set that up with e.g. the Color style (refer to page 67 for details).

Line Element

When inserting a line element you state the coordinates of the two endpoints. The endpoints are inserted in inches and pels and by first stating the distance from the top of the paper edge and then the distance from the left of the paper edge. The attributes of the line element is set via the Geometry style. The example below creates a vertical line which is 2 inches from the left edge of the paper. It starts line 1½ inch from the top and ends 3 inches from the top: (measurements are here in inches and pels).



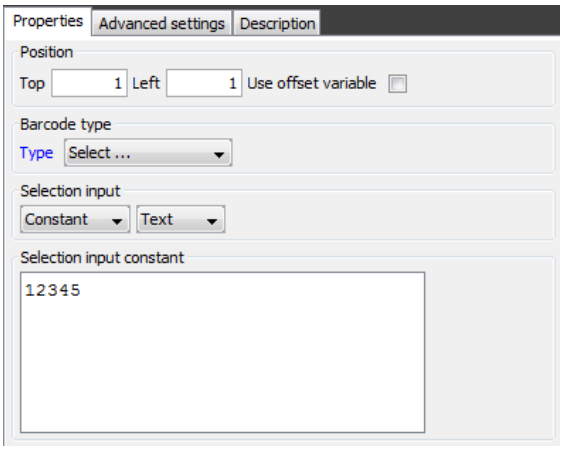
The settings of the line (as well as for the box element) are defined in the geometry style as described on page 64.

Box Element

The box element inserts a box (or frame) in the output. The coordinates and the other settings are set exactly like for the line element above.

Barcode Element

The barcode element works exactly like the text element above:



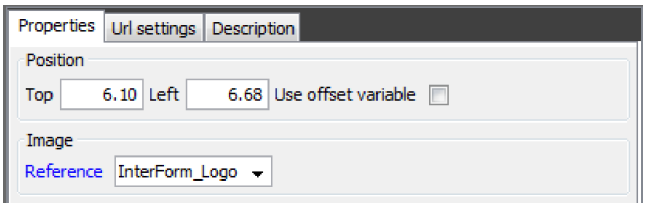
Compared to the text element, there are only 2 differences:
1) The barcode attributes are set via the **barcode style** (refer to details on page 64).
2) The **barcode type** field can be used for selecting a specific barcode type. If this is '**Select..**' like above, then the barcode type selected in the last barcode style is used (or a code39 barcode if no barcode style was found).

Refer to the description of the text element above for additional details.

Image Element

Inserts an image. The image should first be imported as described on page 52. You set the reference point (or alignment) in the Image style as shown on page 66.

The image element simply states the image and where to put it:

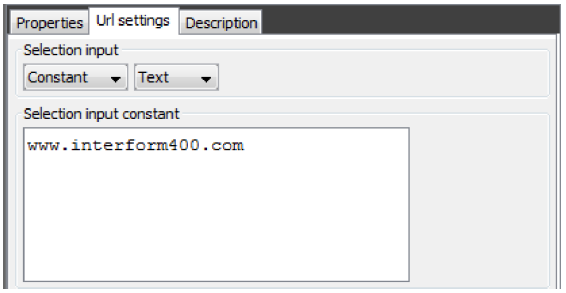


The reference is a drop down list of all images, that have been imported into this template.

If you have just imported the image you might need to save, close and reopen the template before the new image will fully work.
Please notice, that the print style should be set correctly to get the best result when printing. Refer to page 69 for details for this.

The size of the image is set during import. Refer to page 52 for details.

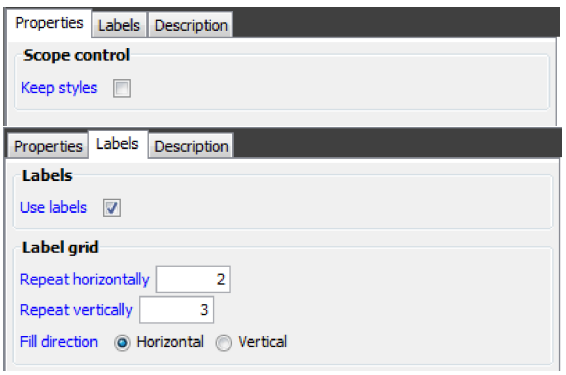
If you click the 'Url settings' tab you will see this:



With this you can link the image with a url. The url does not need to be a constant, but can be a variable or e.g. even an xpath expression, which e.g. selects information from the input XML file(s).

Page Element

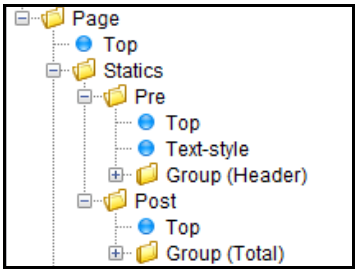
When you insert the page element a new tree in the layout window is inserted. This is where the contents of the page is defined. You insert this to let InterFormNG know that a new page is to be inserted in the output. Unless you only want to export a group you should normally only insert design elements within a Page element. Styles (including a page style) can however be stated before the first page element. The Page Element has these attributes:



Keep styles. If you activate this, then any styles defined in this page 'tree' will stick and also be used after this page. Otherwise the styles defined in the 'tree' will not have any effect outside the page.

Labels. If you activate 'Use labels' you are telling InterFormNG, that there could be labels on this page. You will then be asked for how many labels there should be in both the horizontal and vertical direction. Also the sequence in which the labels are filled out is also defined here. If there is more labels defined than can be fitted into a single page an automatic page break is issued.

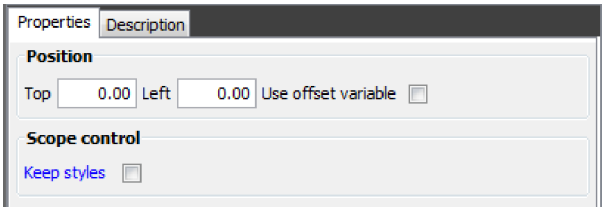
When you insert the page element a subgroup called Static will also be inserted as below:



The Statics group has 2 other subgroups: 'Pre' and 'Post'. The **Pre** subgroup will be executed before the other elements of the page will be inserted, so this will normally be used for the header elements as indicated above. If you use flow you can place the detail lines directly after the header elements. The **Post** subgroup is executed as the last thing before a page break. This can e.g. be used for printing a total, when all detail lines (for the current page) has been printed.

Label Element

The label element is to be inserted within a Page element tree as mentioned above. With the label element you insert multiple labels per page in the pattern as indicated on the page element. These are the properties of the label element:



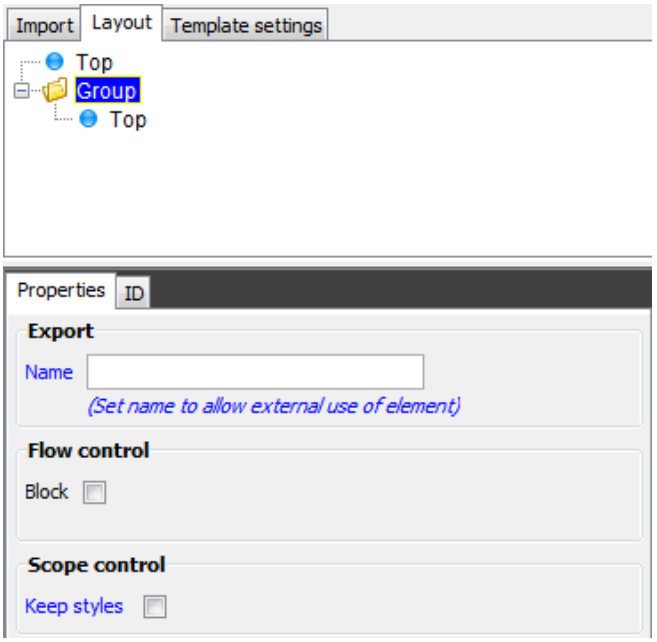
The page element decides for the actual positions of the labels, so the position here is the position relative to this. With the **keep styles** option you tell InterFormNG to keep the styles defined within the label element subtree even after the label - influencing the rest of the page. Remember to set the properties with the label style before inserting a label. The **label style** is mentioned on page **68**.

You need to enable the labels and set properties on the **page element** (in the label tab) and then you can insert the label elements inside the page element. You would normally insert the label element inside a ForEach element (to add a label for each node in the xml file) or e.g. inside a Repeat element (to insert e.g. a constant or calculated number of labels).

Group Element

With the group element you create a subtree or group of element, that can be exported and used in other templates. You can also use the group element to divide your layout into logical sections making it easier to maintain your template. You group elements by placing them in the same group element, This makes it very easy if you want to change the position for them all (or change another style attribute).

You also need to group all elements in the template if you want it all be executed when included e.g. from the main template used for the processor service.



Scope control: Keep styles you can make InterFormNG remembering the styles after exiting the subtree. This is the same as for the Label element above.

Include Element

The Include element includes a section/element from this or another template. In order to do this you will first need to:

1) State an export name in the section/element in the other template.
2) Import the other template in this template.

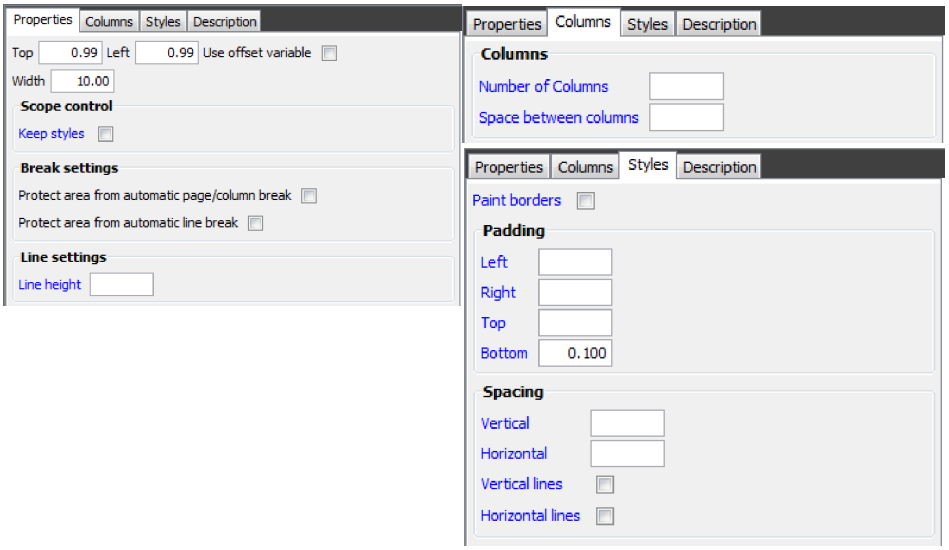
Simply refer to the template and element above. If you want to include an element from the current template, then you just leave the template field blank. Note, that you can even do recursive includes.

Fixed Area

Use the element to group elements physically together. Then you can move all elements inside by moving the complete area. This can also be used inside a flow area when you want to create columns for all the detail lines. Refer to the flow area below for the settings available.

Flow Area

You can make InterFormNG position each element relative to each other with this feature. This makes it possible e.g. to place elements (especially **text**) relatively - both horizontally and vertically as well as insert either line or page breaks when necessary. Use **NewLine** element with this style to position elements relatively below each other.



Top/Left
Start position of the flow area. The flow are will position the element relative to this position and move from left to right relative to the last **text** element inserted with the flow area.

Use Offset variable
If you enable this, then the flow area is positioned via the value of an Offset variable instead of a fixed point as specified above. This can e.g. be used if you want a flexible layout, where the detail lines (flow area) starts in one position on page 1 and in another position on the following pages.

Scope Control / Keep Styles
If you activate 'Keep styles' any styles defined within the flow area will 'stick' even after exiting the flow area. If this is not enabled, then all styles defined in the flow area will only have effect within this area.

Break Settings / Protect area from automatic page/coloumn break
If you want InterFormNG to consider everything within this flow area to be a block (and avoid a page or column break in the middle), then you should enable this option.

Break Settings / Protect area from automatic line break
If this is disabled, then the text will not force a line break - even if it cannot fit within the width set for this flow area. If it enabled the text will be moved onto the next line if there is not room enough.

Line height
The distance set here is the line height when a new line is inserted. This can e.g. be used for keeping a fixed line height even if some of the lines are using a smaller font. (The line height will normally e.g. adapt to the height of the characters used for the text elements and always go below any elements inserted in the flow).

Under the **Columns** tab you have these two settings:

Number of Columns
As default only one column will be added to each page and a page break is issued if a NewLine is added and there is no more room on the page. Alternatively you can here set a number of columns to add for each page.

Space between Columns
If you set a number larger than 1 for the 'Number of Columns' above, then you can here set the distance that you want between each column.

Under the **Styles** tab these last options are found:

Paint borders
Enable this and you will get lines around the borders of the flow area. This can e,g. be used for creating tables or if you temporarily want to see the actual borders.

Padding
A distance between the leftmost/rightmost/top/lowest element within the flow area and actual borders of the flow area. If you enable the border to be painted above, then this padding is clearly shown.

Spacing
If you want additional spacing between each element in the flow area, then you can state it here - both in the vertical and horizonal direction. Enabling horizontal or vertical lines will insert a line between each element. These lines can be used for building up a table within the flow area.

Refer to page 95 for a couple of examples of how to use the flow area.

New line Element

The element, **NewLine** works with the **flow area** (Refer to page 81 for examples) and sets the position of the following elements to be just below the elements, that have just been placed. It will adapt the line distance to the size of the elements, unless you have specified a specific line height in the flow area. If you e.g. change the font size for some texts, then this element will increase the line height.

ColumnBreak Element

Works like the NewLine element except that this element force a column break i.e. force the flow area to continue on a new column (if more than 1 column is defined). If the current column is the last on the page a new page is issued.

PageBreak Element

Works like the NewLine and ColumnBreak elements. The PageBreak element will force a page break if used inside a flow area.

Link Element

With the link element you can embed a file in the PDF files that you create. You can embed any file format. This can e.g. be used for adding a descriptive document for each detail line in the main document like below or e.g. for adding the general terms and conditions into invoices:

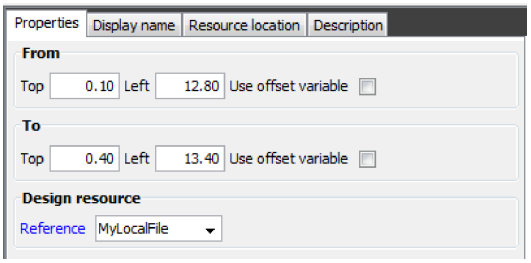
Product	Model	I-Group	SN	License code
Swiftview(50)	525	-	44B4011	D0 44 A9 FF
InterExcel400	515	-	44A3531	F4 5F 13 31
InterForm400	515	i050	4449079	F2 E7 20 AA
InterForm400	810	i100	44B2031	AA BB 01 23
InterForm400	890	i600	44D3971	1A BC DE 44
InterForm400	520	i100	44A1460	79 10 22 F1
InterForm400	530	i400	44A3773	4E FF 01 33
InterForm400	270	i100	44A1597	D0 99 70 07

Data sheet for Swiftview(50)

You open the embedded file by clicking this icon:

You can also specify a display name which is shown when your mouse pointer is set to this position. This is shown above on the right.

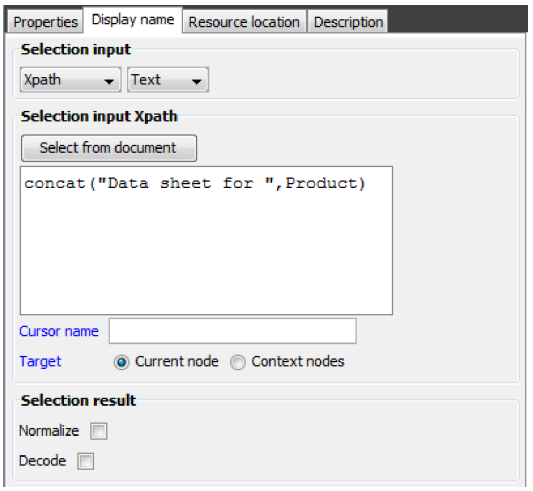
The links/embedded files above have been setup inside a ForEach loop like below:



The positions above specifies an area in which this will be inserted: The clip will fit the size of this area.

As the **Design resource** you can refer to a file, which you have previously imported (refer to page 53). This will be used for previews in the designer instead of the real resource referred to later.

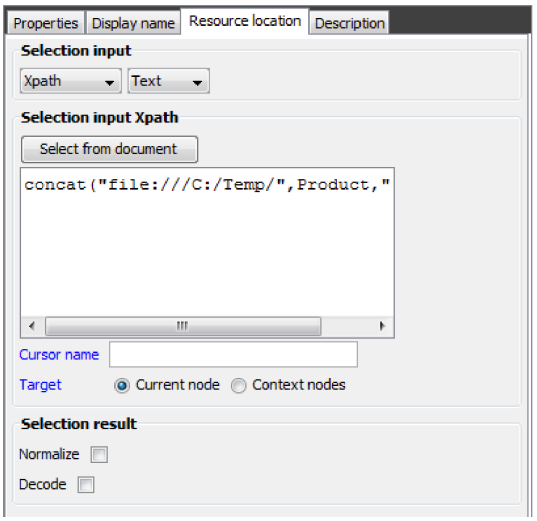
With the 'Display name' tab you can set a text to be displayed, when the mouse pointer is just above this link:



(The result is shown on the previous page as the yellow frame on the right).

In this example we are using Xpath to state a dynamic text, where the fixed text 'Data sheet for ' is followed by a value found in the XML file (Product). Alternatively you can also select a constant, variable, function or even a value from a translation table.

The Resource location tab is for selecting the file to be embedded:



In the case above the complete Xpath command is:

```
concat("file:///C:/Temp/",Product,".pdf")
```

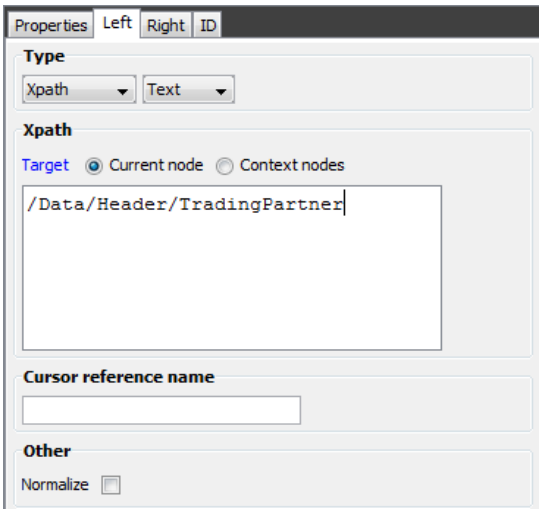
- using the Product field from the XML file as a part of the dynamic file, that we want to include.

If Element

The If element executes a subtree only if a certain condition is met. The properties are these:

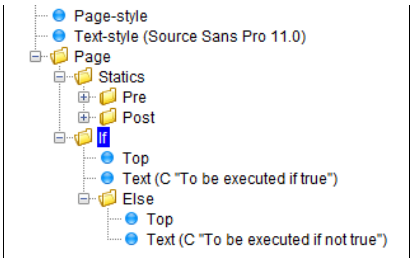
So above you set the datatype (select between Text, Boolean, Number or Date) and set the comparator ('=', '<', '>'....). The **keep styles** option can make InterFormNG remember the styles after exiting the If-subtree, which is necessary if you want to use the IF function to conditionally set a style e.g. the locale style.

Both the left and right side of the comparison can be set like this:



So both the left and right side can be a constant value, a value from the XML file found via Xpath, a variable, a function or a value from a translation table. If you refer to XML, you can also chose to refer to a special cursor (pointing to a place in the XML file) instead of the default cursor via 'Cursor reference name'.

The IF includes an ELSE folder. In case the condition in the IF is not true, then all elements in the ELSE folder are executed. Below we output a text constant depending on if the condition is true or not.



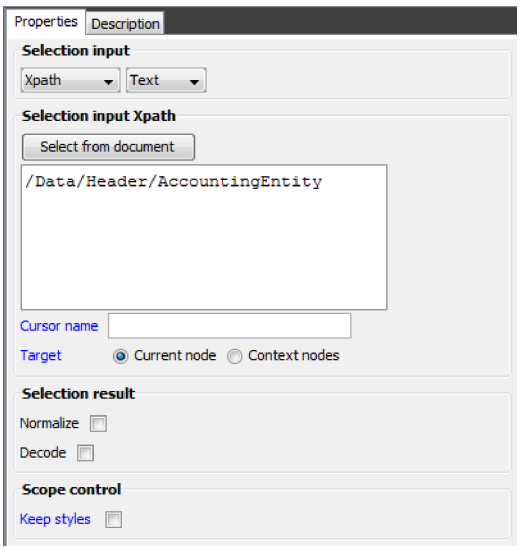
ForEach Element

The ForEach element tells InterFormNG to run a DO-loop for all branches in the XML file with a specific name. With the ForEach Element you can e.g. do a loop for all detail 'lines' in the XML file. The properties are these:

The Xpath is the path to the branches that are to be handled. This is relative to the default cursor - alternatively it can be relative to another cursor - just state it under 'Cursor reference name'. Finally you can with the scope control: Keep styles tell InterFormNG to remember any styles defined in ForEach subtree.

Repeat Element

Use the repeat element to execute a subtree a specific number of times. The number to run can be set as seen below:



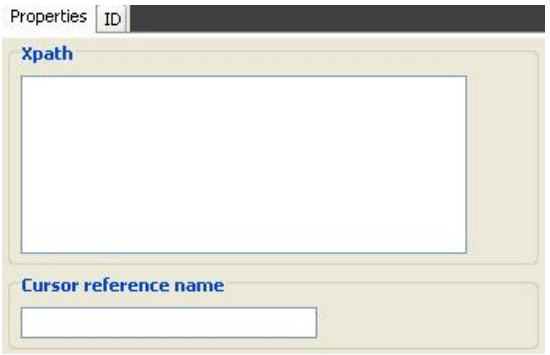
In the example above the number of runs is determined by a number which is found in the XML file to the right. As the number found is 3 the repeat loop is run 3 times. Any styles defined are kept after exiting the repeat loop because of the selected scope control. The number of repeats can also be set to a constant, variable, function and even a translation value.

Define Cursor Element

Use the 'Define Cursor' element to define/create a new cursor (or pointer), which is pointing to a position in the XML file. This cursor can later be referenced in other elements like e.g. Text, If and ForEach. In the properties you simply state the name of the cursor. Refer to page 47 for an example of how this can be used.

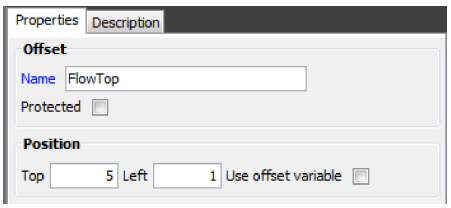
Select Cursor Element

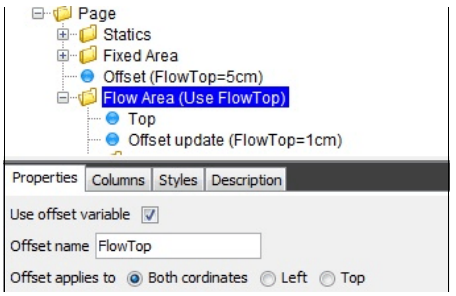
The 'Select Cursor' element sets an existing cursor to a path in the XML file defined by Xpath. State the cursor name in properties unless you want to set the default cursor. The cursor can later be used in other elements when referencing to XML file data (then you just need to refer to the relative path from this cursor).



Offset

Normally the design starts from position 0,0 from the top and left and elements are positioned relative to this. Instead you can also name a specific position on the page and then place elements referring to this named offset. An advantage with this function is that you can then later reposition elements by changing the value of such an offset.

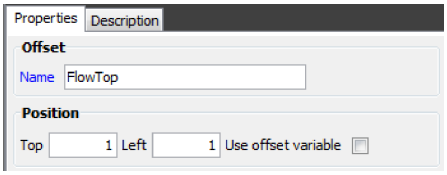




On the right you see an example of how the Offset can be combined with a flow area to have different sized headers on page 1 and subsequent pages in the output: The offset has one value when entering the flow area (to be used for page 1) and is immediately change to the size of the subsequent pages.

Offset update

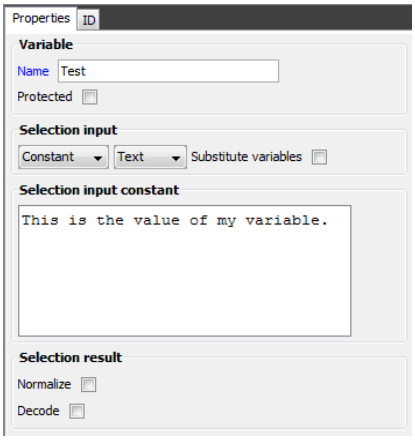
With this element you can change the value of an existing offset already defined:

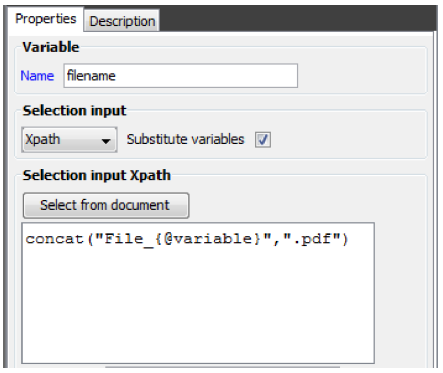


Refer to the Offset element above for an example of how to use this.

Variable Element

With this element you declare/create a new variable element. The variable name and value is set like shown below, where the variable, Test is assigned the value 'This is the value of my variable':

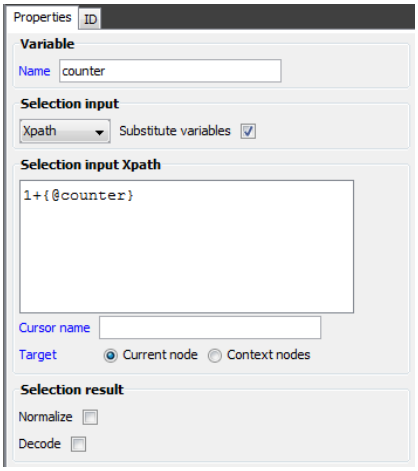




The example on the right defines the value of the new variable based on another variable. Notice, that you need to use Xpath when referring to a variable - both for this and the variable update element.

Variable Update Element

This element is used for updating an existing variable. You use this element to to change the value of an existing variable. The variable element above is used for creating new variables. Here is an example (left): Here we use Xpath to assing a new value to the variable called counter, simular to this:
counter = counter +1



Refer to page 125 for another example using a longer Xpath expression.

Metadata Element

The metadata element is used for adding document metadata information. Some of the possible metadata fields are shown on page 103. The metadata can e.g. be used for specifying what kind of output(s) the incoming XML file should be converted to.

Combining Styles and groups

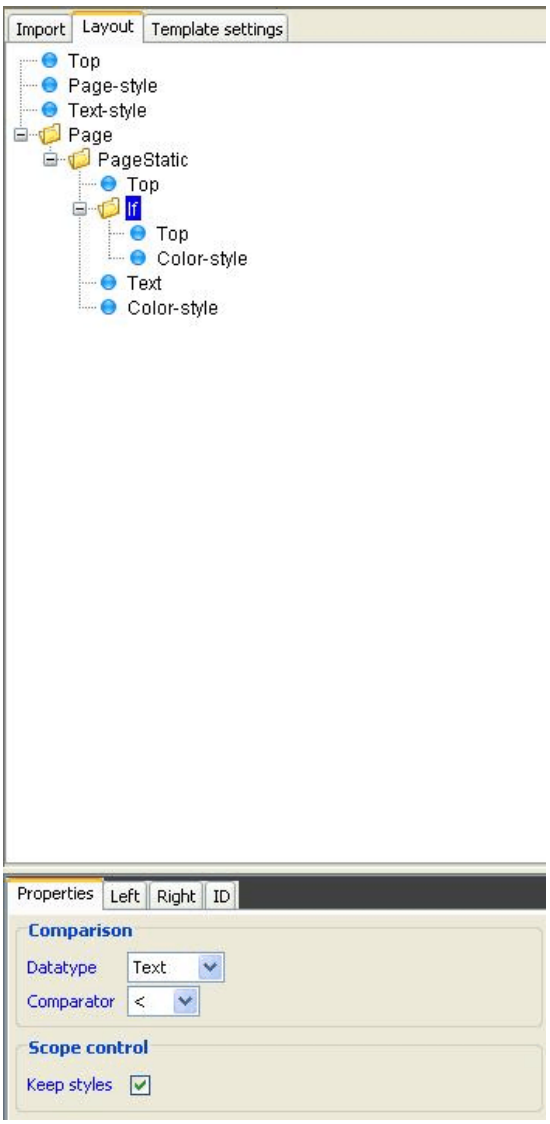
When you insert a style element, then this element is normally only active within the current group. A group is defined as a subtree, which can be created by inserting one of these elements:

Page element
Label element
Group element
Fixed Area
Flow Area
IF element
ForEach element
Repeat element

This means, that unless you activate the ‘Keep styles’ option of the surrounding element, then the style will be forgotten when leaving the actual subtree.

Here is an example:

We want to output an amount from the XML file, but it should be written in red, if the amount is less than 0. We can do that in this way:



Please note, that the text element is outside the scope of the If element, so that it is always inserted in the output. The If element checks if the amount is less than 0 and importantly: Activates: **Keep styles**. This makes the color style within the **IF** scope (which sets the color to red) active - even after exiting the If scope.

That makes the new pen color red (for the text element) if needed. After the text element the color is set to black - setting the color back if it was previously changed.

Other important icons and design settings

Apart from the elements and styles there are a few other icons, that can help you during the design process.

On the very left of the screen you have these icons:

Press this icon to move the marked element upwards in the list.

Press this icon to move the marked element downwards in the list.

Deletes the marked design element. You can also just mark an element and press the <Delete> key.

On the top of the screen these icons can be found:

Press this icon to create a new template.

Opens an existing template for editing.

Saves the current template.

Closes the opened template.

Click this icon (on the upper right) to preview the resulting design in PDF.

You can also use the normal clipboard option cut/copy/paste to change the elements as well as ‘undo’ and ‘redo’ the last changes.

All elements are placed between ‘*Top*’ and ‘*Bottom*’. Activate the position between ‘*Top*’ and ‘*Bottom*’, where you want to insert the element. The new element will be placed after this position.

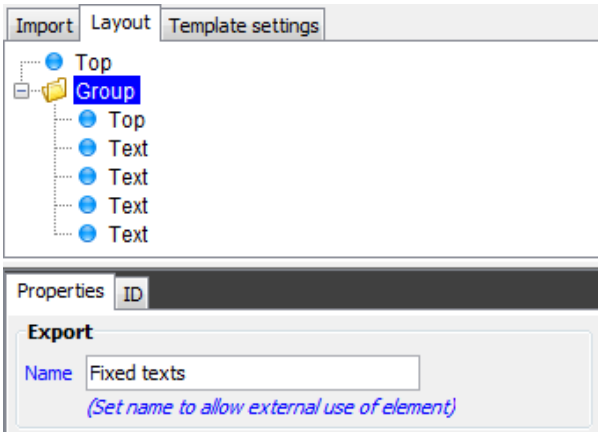
Design examples and tips

When you design your templates it is very often worth while to really consider the structure of the template, that you are designing. Before you design anything you should consider the tips below:

Reuse groups

Are there any groups in existing templates, that could be reused in this template?
(If yes you should of course make them available by exporting them in the other template and import them in the current template).

You do that by first giving the group an export name in the other template:



Now other templates can refer to it. Before we can reuse the group in the other template we first need to import it as described on page 50.

Finally you can use the include element to refer to this group by inserting the include element:

- and then state the template and element to use.

The type of elements that you want to reuse could of course be any element. Just remember, that if you want to share and reuse styles via such a group, then you need to activate 'Keep styles' for the group in question. If you forget the styles defined are forgotten after the group have been executed.

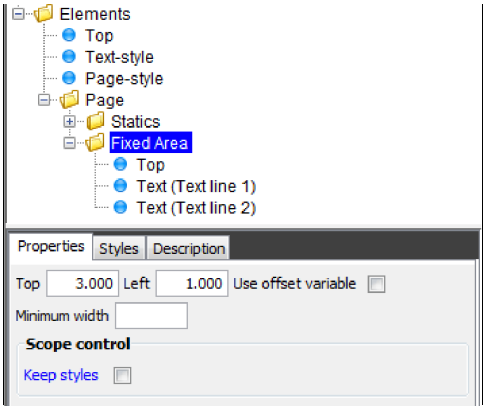
Are there any groups of elements in the current template, that could be reused more than once? If yes: Group the elements together and give the group an export name. Now you can refer to these groups with the include element as well. (You do not need to import groups that are within the same template).

Create Fixed areas when possible

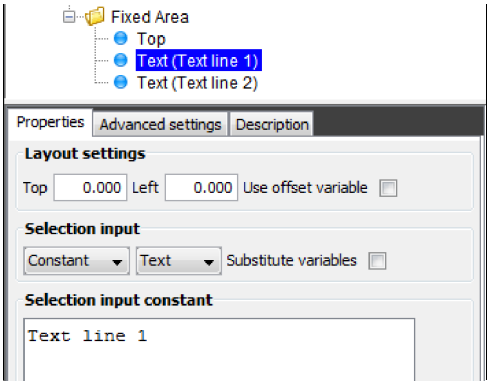
Try whenever you can to group elements together in fixed (or flow) areas. It makes it much easier to maintain in the longer run. It is e.g. much easier to reposition a complete area instead of repositioning each individual element inside. Remember, that you can even create sub areas within areas.

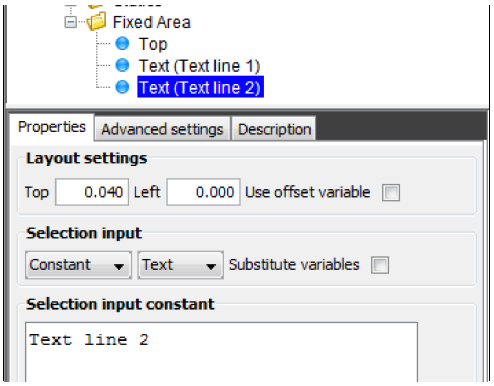
With the fixed area below we state where the group of text elements is to be placed on the

output:



And the first text element is placed as starting in position 0,0 and relative to the fixed area:



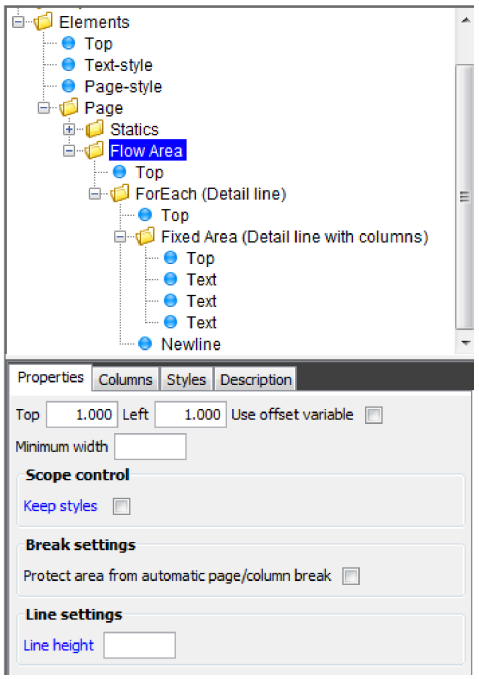


In the final output the 2 texts are printed under each other 3 inches from the top and 1 inch from the left.

The Flow area

Consider the Flow area, if you are handling a very flexible design, where you want elements to be placed relatively to each other. Such an example are detail lines. Here you can even use the NewLine element to force a line break.

With a setup like below we are printing all the detail lines of an xml document:



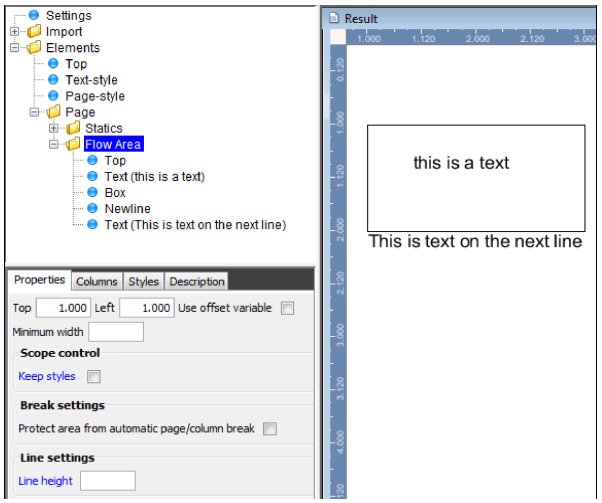
The flow area will position the detail lines starting 1 inch from the top of the paper and 1 inch from the left.

The ForEach will loop over all the detail branches found in the XML file.

Each detail line is to presented as nice columns, so that is why a Fixed area is used.

The NewLine element will make the output advance one line for each detail line - within the flow area only. The term ‘line’ should here be thought of as any element or even a group of many elements (not just text elements). Use the NewLine element to advance the flow to below the elements inserted - unless you specify a specific line height on the flow area.

An example of combination of multiple elements and the NewLine element can be seen below:

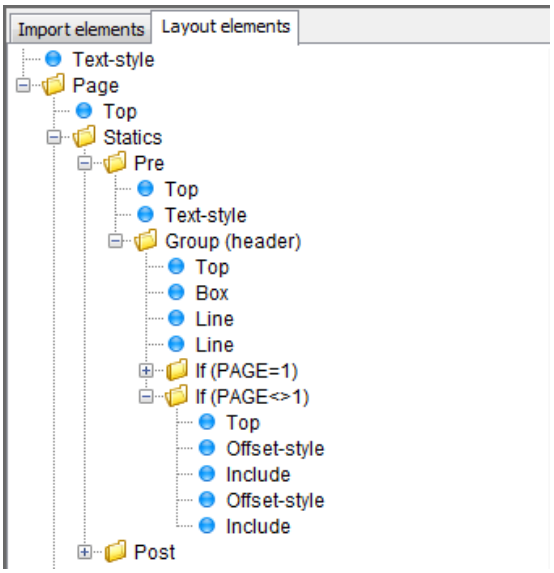


The Page Static group

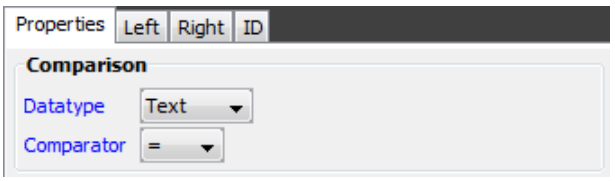
Consider the design you want: Are there any elements, that should be included in fixed positions on each page (e.g. a header, a footer or even a watermark) ?

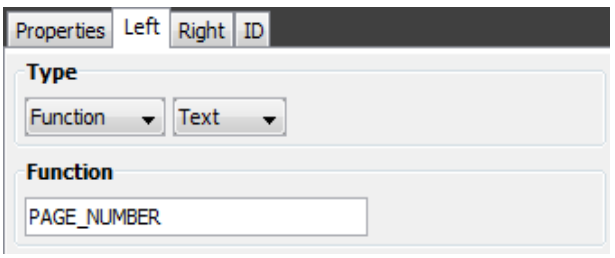
If so it should be considered to place these elements inside the static group of the used page element. Elements inside the static group (of the page element) are executed automatically on each page. Here we even use the function, PAGE_NUMBER for conditioning the elements based on the current page number - with the If element. In this manner we can create one header for page 1 and another for the other pages.

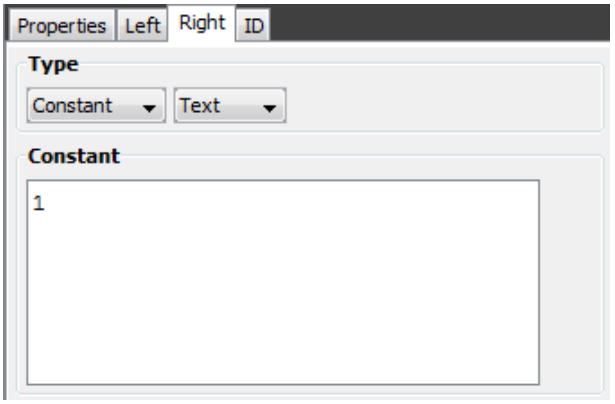
An example of such a setup is shown below:
Notice, that the 'Pre' subgroup under the 'Statics' is executed before the page is filled out (stated outside Statics), and the 'Post' subgroup is executed after the page is filled out (and a page break is executed right after this). The 'Post' can e.g. be used for triggering a page total or simular.



The If conditions look like this:

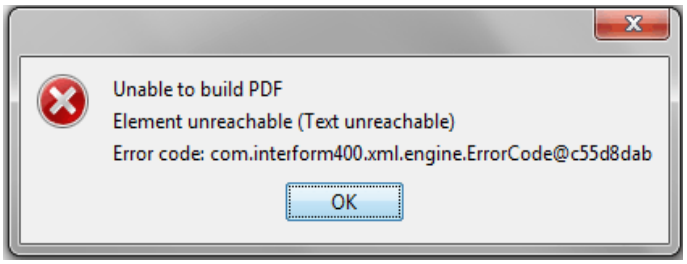




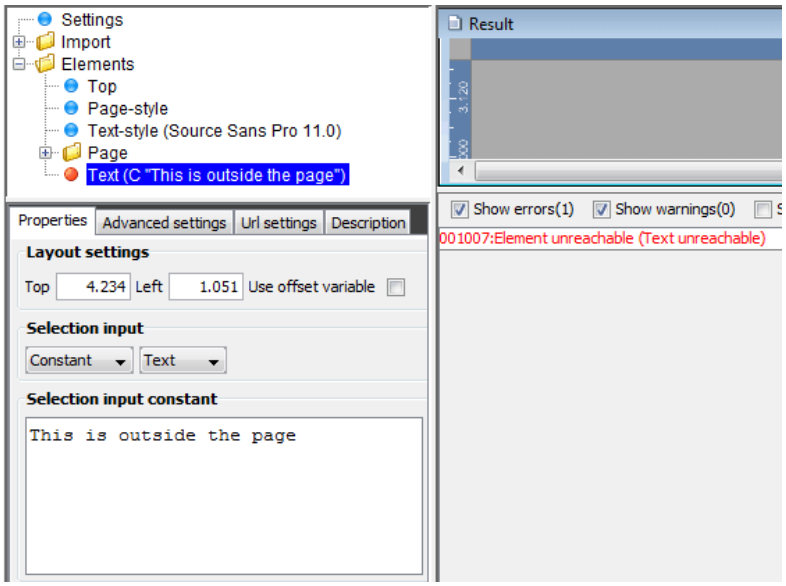


Solution to: Element unreachable

If you get an error like this when trying to preview the result of your template:



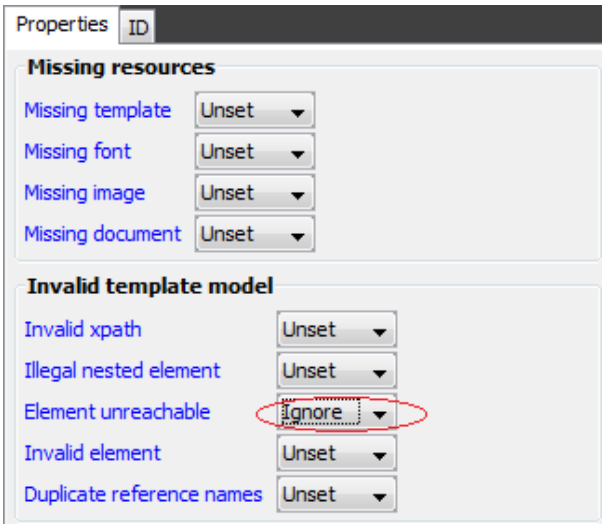
Then you have inserted a design element outside a page element e.g. like below:



The error message indicates, that at least one design element is placed outside a page element like above. (The text element should have been placed within the scope of a page element). You can verify if it is outside by minimizing the Page folder (Clicking on the '-' next to Page): If the text element is not shown, then it really is inside the Page element.

Normally you will get this error because you have misplaced the design element, but in some situations you might want to do this, if you e.g. want to build up one or more groups (to be reused in the design). Such a template could contain a range of groups offering predefined designs that you could use in all templates in the company.

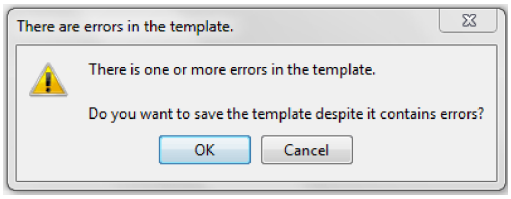
If you want such a setup, you can actually tell InterFormNG to ignore this error (if it should occour) by inserting an error style with this setting:



So set ‘Element unreachable’ to ‘Ignore’ and then you will not get the error above.

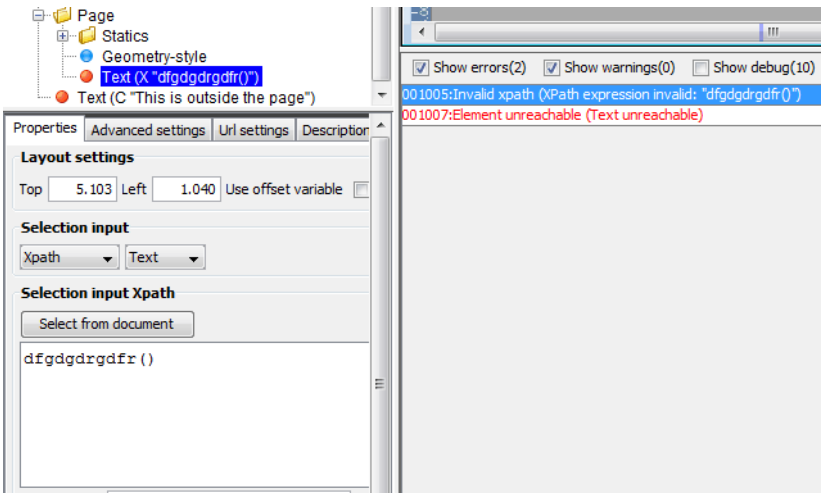
There are errors in the template

When you save a template this window might pop up:


There are errors in the template.
There is one or more errors in the template.
Do you want to save the template despite it contains errors?
OK
Cancel

By clicking ‘OK’ you can chose to save the template in spite of any errors. If you want to identify the exact errors, then you can click this icon on the lower right of the designer to list all errors:

Then you can click each error in the list on the right to highlight the corresponding element in the list on the left:



Setting up InterFormNG for automatic conversions

After you have setup one or many templates you can now setup InterFormNG to automatically convert new XML files into PDF. You do that via Document Processor service.

Installing/Uninstalling the Services

If you have not installed the services during the installation of InterFormNG you can also install them later by selecting this in Windows:

Select: <Start>, 'All Programs', 'InterFormNG', 'Service' and 'Install Service'.

You uninstall the services by selecting:
<Start>, 'All Programs', 'InterFormNG', 'Service' and 'Uninstall Service'.

Configuring the Services

Before you start the service you should determine the source directory (of the XML files to be converted), the output type and destination and also what template to use for the conversion.

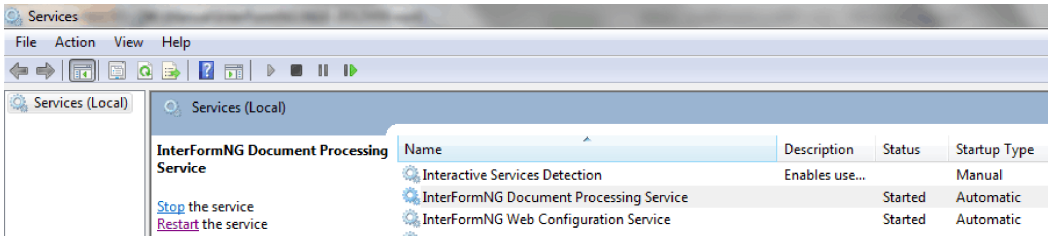
Even though you currently can only refer to one template for the conversion you can still use conditions in the template to use other templates depending on the contents of th incoming XML file.

Refer to page 25 to see how you can configure the services via the web interface.

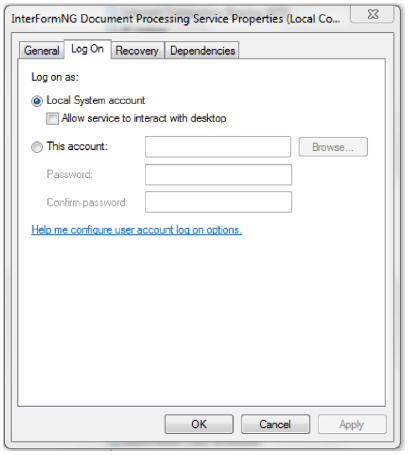
Remember to always restart the Document Processor service to activate any changes of the processor configuration.

In some installations you might want to run the services under a specific user (e.g. an administrator). If you want to do that you can set the user in this manner:

In windows go into the Control panel and select 'Administrative Tools' and then 'Services':



Now double click on the service in question and after selecting the ‘Log on’ tab you can set a specific user to use for this service:



Starting and stopping the Services

After installing and configuring the service you can now start and stop the service by selecting this in Windows:

Starting the Document Processor service:
<Start>, ‘All Programs’, ‘InterFormNG', ‘Service’ and ‘Start Service’

Stopping the Document Processor service:
<Start>, ‘All Programs’, ‘InterFormNG', ‘Service’ and ‘Stop Service’

Starting the Web service:
<Start>, ‘All Programs’, ‘InterFormNG', ‘Service-web’ and ‘Start Web-Service’

Stopping the Web service:
<Start>, ‘All Programs’, ‘InterFormNG', ‘Service-web’ and ‘Stop Web-Service’

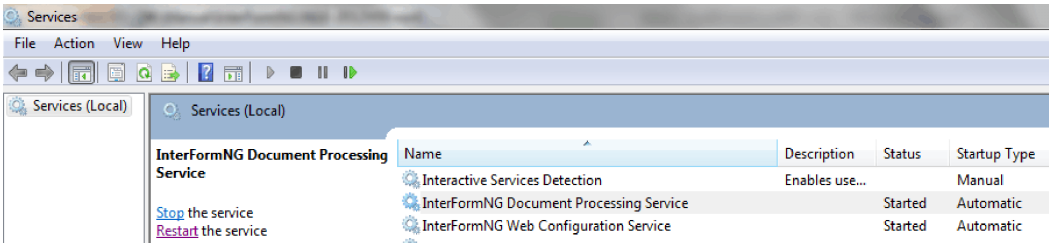
A DOS window may briefly be opened and closed to indicate that the request is being executed.

Alternatively you can start, stop and restart these services from the Windows Control panel in this way:

1. Go into Control Panel.
2. Select ‘Administrative Tools’ and then ‘Services’ or e.g. type ‘Services’ here:

- or select ‘Start’ and type in ‘Services’ here:

3. Now you can chose to stop, start or restart both InterFormNG services:



Using the Document Processor Service

You can envoke the Document Processor service in a few ways:

1. You can make the processor monitor a directory for new XML files and make it react whenever a new XML file arrives here.

2. You can include the XML file as a field in a database and use a JDBC driver to make the processor react to that. Refer to page 148 for details.

3. You can use the workflow in combination with both options above to configure how the XML files should be processed. The workflow setup is described on page 109.

4. Instead of the workflow you can also let the default template of the processor branch out to the relevant template and itself set all necessary meta data to specify the output that you want.

5. You can also call the processor via a socket call e.g. for previewing the result of a merge between the XML file and the default template. The socket call is described on page 140.

Below the combination of option 1 and option 4 is described in details.

In order to use the service you simply move or copy the XML files onto the xml_inbox as configured above. If the service ist started it will automatically convert the file and either print, create a pdf file or email - depending on the configuration and the meta data setup in the template.

When the incoming XML files are processed they will be moved: First they are moved to the subdirectory called ‘Processing’ and after processing the files will be moved to either the ‘Processed’ or ‘Errors’ subdirectory depending on if the merge when OK or not.

Setting up automatic printing

When a PDF document has been produced, InterformNG allows the document to be printed on any printer that supports PostScript or PCL5 printing. Label printing on ZPLII compatible printers (normally Zebra printers) is also supported. The meta data key called ‘mediaType’ should be set to ‘print’ in the used template in order to enable printing and you should normally select the printer by stating a documentType (meta data) in the template and linking the documentType (meta data) with a printer in the server configuration.

All meta data keys are listed below.

Refer to page 35 for the printer setup of the server.

Setting up automatic PDF creation

If you want to create PDF files you need to enable this option in the web interface and set the meta data 'mediaType' to 'archive'. The path to state where the PDF files will be created is combined of 2 parts:

1) The 'root' directory within all PDF files will be created. This is setup in the server via the web interface.
2) The directory above will be combined (or concatenated) with the path stated in the meta data key 'interform.plugin.archiver.folderName', which you should assign in the used template.

The output file name will be the default name setup in the web configuration unless you have specified the file name via the meta data key 'interform.plugin.archiver.fileName' in the template.

The setup of the pdf attributes for the server is setup as shown on page 36.

Setting up e-mailing

E-mailing from InterFormNG require, that you setup the emailing (refer to a mail server) in the web config e.g. as shown on page 37. Here you can also state default subject and the default contents (defined in an **.rtf** or **.html** file). Both of these can however be overridden by assigning values to these meta data keys: **email.subject** and **email.templateName**. Notice also, that the email template (rtf or html) can refer to meta data set in the template. You can also refer to an email local via the 3 meta data keys for email.locale as shown below. They should be added to the email template name in the same sequence as listed below: email-template_language_country_variant.rtf.

You can even define your own meta data keys in the template and let the email template (rtf or html file) refer to the meta data in this manner:

In the .rtf or .html email template:

Dear @[receiver.fullname],

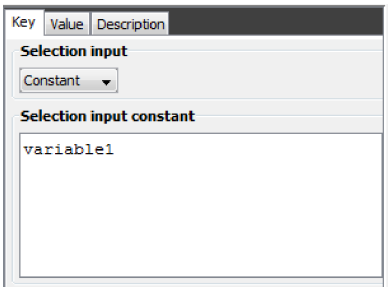
According to our agreement we now forward you a test email.
*This is variable1 from the template=**@[variable1]***

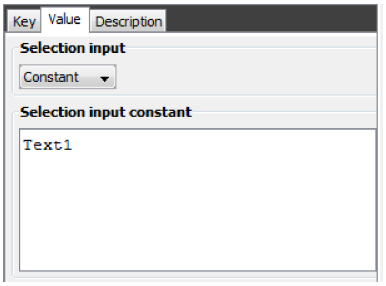
E-mailed from InterFormNG

Best regards,

Peter Sørensen
InterForm A/S

Notice the bold text above, where we refer to the **meta data key**, variable1 which is defined like this in the template:





The final email contain this:

Dear Kim Egekjaer,

According to our agreement we now forward you a test email.
This is variable1 from the template=**Text1**

E-mailed from InterFormNG

Best regards,

Peter Sørensen
InterForm A/S

The email templates must be stored in {INTERFORMNG_HOME}/resources/mail-template or a subdirectory inside this.

Selecting the output via metadata

You can let the template use information from the XML file to set the actual output e.g. depending on the contents of the XML file. The metadata element looks like this:

(Refer to page 89 for additional information)

The most important metadata fields are these:

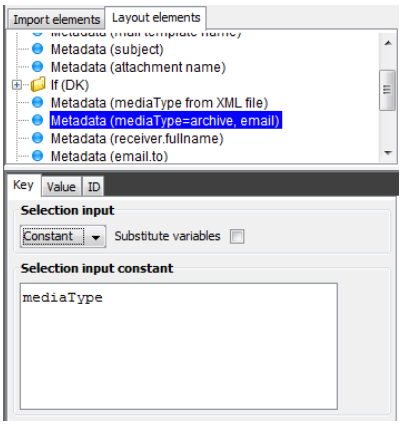
General metadata

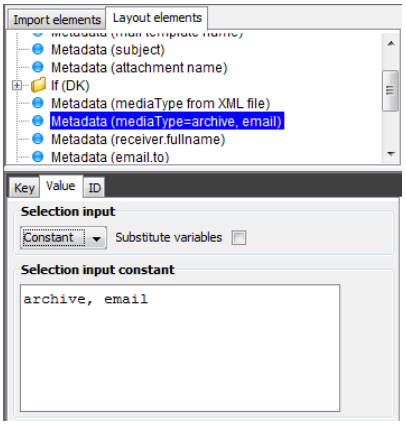
These are the metadata, that can be generally used for all outputs:

mediaType This metadata determines the output that is wanted. It can be set to these values:

print	Prints the result on a printer.
email	Emails the pdf file, that is the result of the merge with the template.
archive	Archives the result as a PDF file.

If you want multiple outputs you can assign multiple values separated with a comma like this:





(In this case the XML file will be merged with the template and saved as a PDF file and after that it will also be emailed).

multiup Possible values are **true** and **false**. (False is default).
If you want to place multiple 'forms' or pages on a single page of output (often also referrred to as N-up, Multi-up or 'pages per side'), you should normally use the label feature of the template as described on page 78.

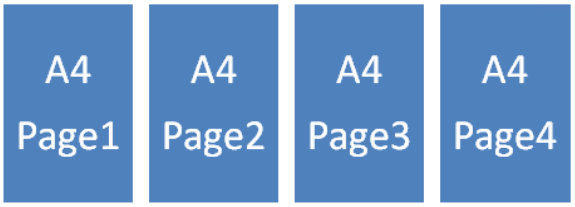
Prerequisites
The multi-up feature has been built especially for the PCL output. Currently it also works for PDF and ZPL output, but the result for PDF and ZPL cannot be guaranteed. For this to work you need to design a portrait design (where the width is smaller than the length of the media). You need to respect the non-printable margin in PCL which is around 6 mm.

This multiup feature makes it possible to reuse the exact same template for 2 different sizes of output.

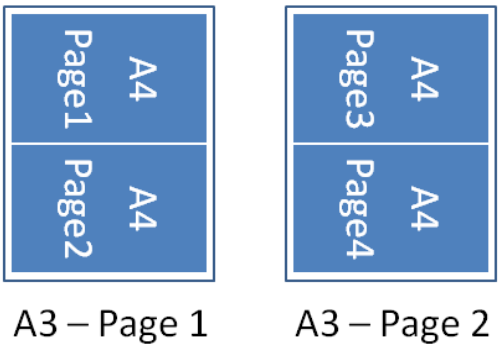
If you set this option to **true**, the output will place 2 pages on one page in the final output and the output size will be the double of what you have defined in the template. The template must be designed in portrait and the height of the page must be equal to or larger than the width. You can set the meta data key in either the template or in the workflow.

The option will move to a new output page if there is a change of paper size between an odd page to the following even page.

Examples: If you set do not change this or set it to **false** your output will be like defined if you e.g. are creating 4 pages of A4 pages in your template:



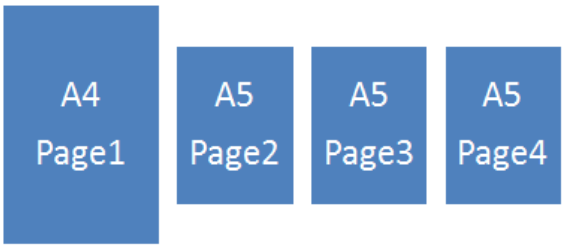
If you now set multiup to **true**, then the output will change:



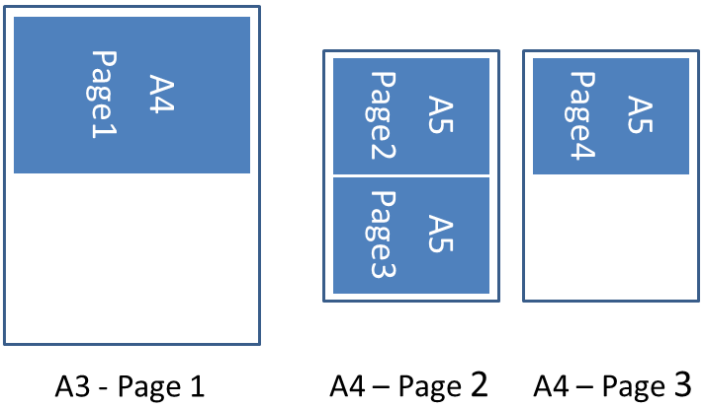
You now get only 2 pages of output, where the original pages are rotated and inserted in the output page, which is now twice as large as the original (Here: A3 instead of the original A4).

If we consider a setup where the page size change, then InterFormNG will adapt to the new size e.g. like below:

multiup=false:



multiup=true:



A3 - Page 1 A4 – Page 2 A4 – Page 3

As you can see above InterFormNG will change to a new output page if the input page size should happen to change out of sequence with the page breaks of the output (i.e. if the page size changes from an odd page to the following even page).

Email metadata

These are the email metadata available:

email.to	The name or names (comma separated) of the recipient(s) to receive the e-mail.
email.templateName	The (file)name of the template to use for the contents (body) of the mail, relative to the mail-template folder. This can be either an html or rtf file. See above for more information.
email.attachment	The name of the attachment to create. You can use this to change the somewhat technical attachment name generated by default to something more meaningful, like "q4 report.pdf".
email.locale.language	Indicate which language you would like the mail to use. The plugin uses this value to select a language specific template. Example: 'en' for english.
email.locale.country	Indicate which country you would like the mail to use. The plugin uses this value to select a country specific template. Example: 'US' for The United States of America.
email.locale.variant	Indicate which country you would like the mail to use. The plugin uses this value to select a country specific template. Example: 'aaland' for a Swedish variant used in Finland.
email.subject	Allows you to specify which subject the e-mail should have.
email.from	Allows you to specify which sender the e-mail should appear to come from.

Print metadata

These are the print metadata available:

interform.destination.printer.printerName
Force a named printer to be used. Normally print is sent to a printer according to the rules defined in printConfig.xml, which is a configuration file belonging to the print manager plugin. Following these rules, a printer is selected based on what document type is being printed. However, it may be that in some cases a template designer wants to force the print to go to a specific, named printer, regardless of any rules defined other places. Note: To avoid having to modify the template when the name of a printer changes, we recommend using the metadata key interform.destination.printer.printerId in stead.

interform.destination.printer.interactive
Possible values: "true", "false"
Display an interactive print dialog prior to printing a document. If the end user needs to be able to manually modify print attributes, a dialog similar to the one found in typical word processors may be displayed to the user, allowing any value to be changed. Note that when InterformNG runs as a service, it is undesirable to enable interactive mode, since the runtime environment may prohibit such dialogs from being shown.

interform.destination.printer.doPrint
Possible values: "true", "false"
Enable printing, yes or no. If this value is set to false, the document will not be printed.

interform.destination.printer.sides
Possible values: "one-sided", "two-sided-long-edge", "two-sided-short-edge"
Specify how pages are printed. Typically this value is derived from the document type according to rules defined in printConfig.xml.

interform.destination.printer.copies
Specify the number of copies to print. Typically this value is derived from the document type according to rules defined in printConfig.xml.

interform.destination.printer.printerId
Select a printer based on id, as defined in printConfig.xml. By selecting a printer by its id in printConfig.xml rather than by specific name, a printer may be renamed without breaking all the templates that refer to it using the metavalue interform.destination.printer.printerName.

interform.destination.printer.pcl.image.fullcolor
Possible values: "true", "false"
Specify if the PCL output should be in color or black and white. Default value is true meaning that the output is in color. Most black/white printers are able to convert the color images in to black/white so the default value should normally be fine. With 'false' the output is in black/white and you should also expect a smaller output data stream with this value. With the default value (true) you might see a problem when printing on older black/white printers, so for those printers you need to set this to false.

interform.destination.printer.mediaSize
Possible values: "iso-a4", "executive", "ledger" etc.
Force selection of a specific paper size. Typically this value is derived from the document type according to rules defined in printConfig.xml. Allowable values vary from one runtime environment to the next. The printConfig command line tool can be used to produce a complete list of media sizes for each printer.

interform.destination.printer.mediaTray
Possible values: "top", "bottom", etc.
Force selection of a specific printer tray. Typically this value is derived from the document type according to rules defined in printConfig.xml. Allowable values vary from one runtime environment to the next. The printConfig command line tool can be used to produce a complete list of media trays for each printer.

interform.destination.printer.orientation
Possible values: "portrait", "landscape", "reverse-landscape", "reverse-portrait"
Force print to be output in landscape or portrait mode. Typically this value is derived from the document type according to rules defined in printConfig.xml

interform.destination.printer.pcl.pcl
With this meta data you can add extra PCL commands in the begining of the PCL print out from InterFormNG. You can e.g. use this to select a specific paper drawer on the printer (e.g. to select between plain paper and preprinted paper.).

The best way to select the drawer is to set the paper type of each drawer on the printer and then insert a PCL command to select the same paper type.

Here is a list of the available paper types:

Paper type	Command
Plain	EC6WdPlain
Bond	EC5WdBond
Color	EC6WdColor
Labels	EC7WdLabels
Recycled	EC9WdRecycled
Letterhead	EC11WdLetterhead
Cardstock	EC10WdCard Stock
Prepunched	EC11WdPrepunched
Preprinted	EC11WdPreprinted
Transperancy	EC13WdTransparency

Please notice, that the PCL commands must be written exactly as above. They are case sensitive and the space between 'd' and 'S' in 'EC10WdCard Stock' must be included.

Notice, that even though the printer menu might show each of these printer types in your local language you should still use the english command shown above.

If the printer does not support selection of the drawer via the paper type, you can use an alternative command: EC&l#H, where the # is a PCL drawer number. So if you e.g. want to select PCL drawer 5 the command is: EC&l5H.

Please notice, that the PCL drawer number can range from 1 to 255 and is NOT the same number as the physical drawer on the printer and that the number depends on the exact printer model used. For some printer you even need to select drawer numbers in the range 250-255.

If you want to insert multiple PCL commands you just concatenate them into the same meta data.

interform.destination.printer.pcl.pjl
This meta data key works like **interform.destination.printer.pcl.pcl** except that this command inserts a PJL command in the begining of the PCL print out. This PJL command can e.g. be used to add extra PJL commands to call functions on the printer.

interform.destination.printer.finishings
Possible values: "none", "staple", "cover", "bind" etc.
Specify finishings, such a stapling. Typically this value is derived from the document type according to rules defined in printConfig.xml. Note that your printer may support many different values for this attribute, according to its physical capabilities. The printConfig command line tool can be used to produce a complete list of finishings values for each printer.

interform.destination.printer.zebra.clearBuffer
Possible values: "true", "false".
This value is specific to Zebra label printers. If the value is true, the system will output a ~JA command immediately before printing the label, thus clearing any previous print jobs that the Zebra printer may have queued.

The InterFormNG WorkFlow

You can do automated merges like described in the previous section, but InterFormNG also contains an alternative which may help you for more complicated setups. This is called WorkFlow. With the workflow you can select specific XML files from name, directory or contents and the specify one or multiple actions for the XML files fitting the condition.

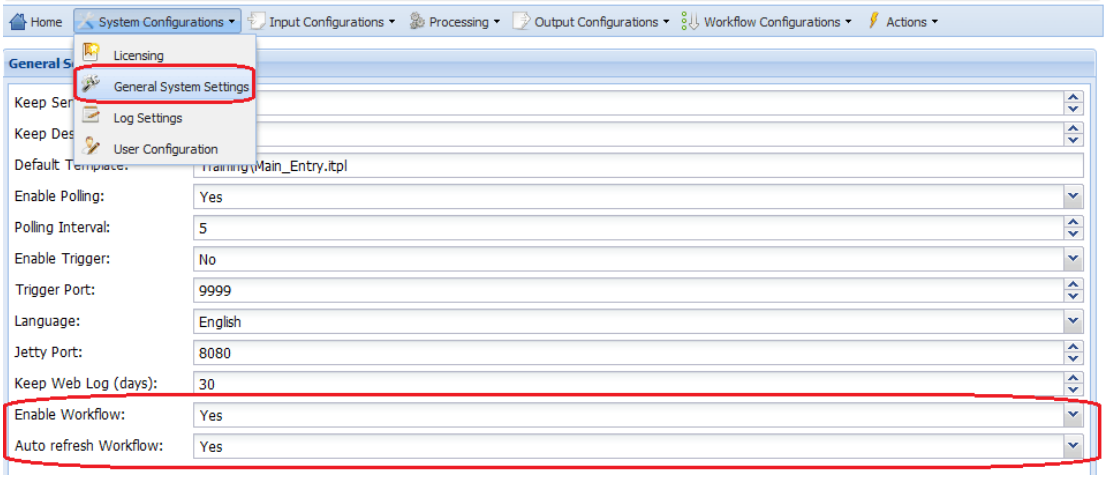
In general the workflow settings overrules the meta data settings, that you may have done in the used templates - when activated. You can setup the workflow via the InterFormNG web interface.

The workflow consists of 2 elements:

Template selections and **Output configurations**. The **Template selections** sets the template and what kind of output you want and the **output configurations** are only needed if you want to override the printed output to a specific printer (or specific printing features) after running the template.

Enabling the Workflow

In order to use the Workflow you should enable it. It is a good idea first to setup the Workflow so that the service will run as before and then enable the WorkFlow. The WorkFlow is enabled from the web interface by selecting 'System Configurations' and 'General System Settings':

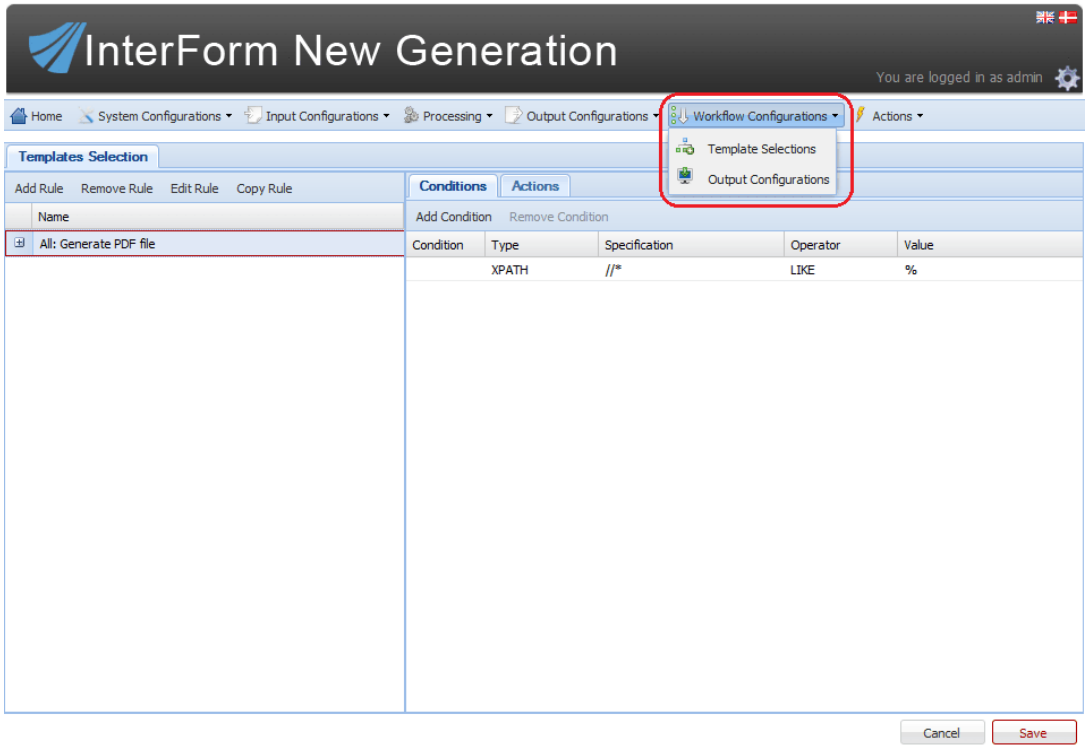


In the bottom of the screen you can enable the Workflow and enable Auto refresh for the Workflow. If Auto refresh is enabled you will not need to restart the InterFormNG Processing service to put workflow changes into production.

Remember to restart the processor service after enabling/disabling the workflow - as shown on page 100.

Working with Template Selections

Select 'Workflow configurations' and 'Template Selections' in the top to see all the rules:



A rule is a selection of XML files. When a new XML file arrives in the XML_inbox directory as defined in the configuration, then the Workflow (if activated) will compare the attributes and contents of the XML file with each of the rules listed until a condition for a rule is validated as true. Only the first true rule is executed, so the sequence of the rules is quite important. You can rearrange the rules simply by dragging a rule up or down in the list.

(For each rule you can later setup one or more actions, but we will come back to that later).

When adding a Rule you are prompted for a name. After entering the name you can on the right specify the condition for the XML file selection:

The condition can be built up with these:

Condition: You can combine conditions with either '**AND**' or '**OR**' for this. The first line for a rule should be blank.

Type: This is the type of condition you want to use. Possible values are:

XPATH Use this to condition the rule on the contents of the XML file. Via the Xpath language you extract information from the.

DIR Select the rule by comparing the directory where the XML file is found.

If you set the operator to **EQUALS**, then the full path must be specified e.g. like **C:\XML_inbox\processing**.If you state **LIKE** then you should specify a unique directory in the path - without any '\' e.g. **XML_inbox**.

FILENAME The file name of the XML file can be used for conditioning the rule.

METADATA When calling InterFormNG either via the database plugin or via a socket call, you can set one or more metadata keys. With this you can condition the rule on these metadata keys.

Specification: If you select XPATH for the type above, then you should insert the Xpath expression here. For other types this should not be used.

Operator: Possible values are:

LIKE	Here you can use a compare text as the value using various special conditions for the comparison - se Value below.
EQUALS	The expression selected in type (and specification) must be exactly what is specified in the Value field.

Value: Here you can specify either a fixed string - if you are using 'EQUALS' as the Operator or a compare expression if you are using 'LIKE'. For the LIKE option you can use these features:

%	Signifies any string which is has a length of 0 or more.
\\	The backslash character
[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z](subtraction)
.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

Other valid expressions can be found via this link:

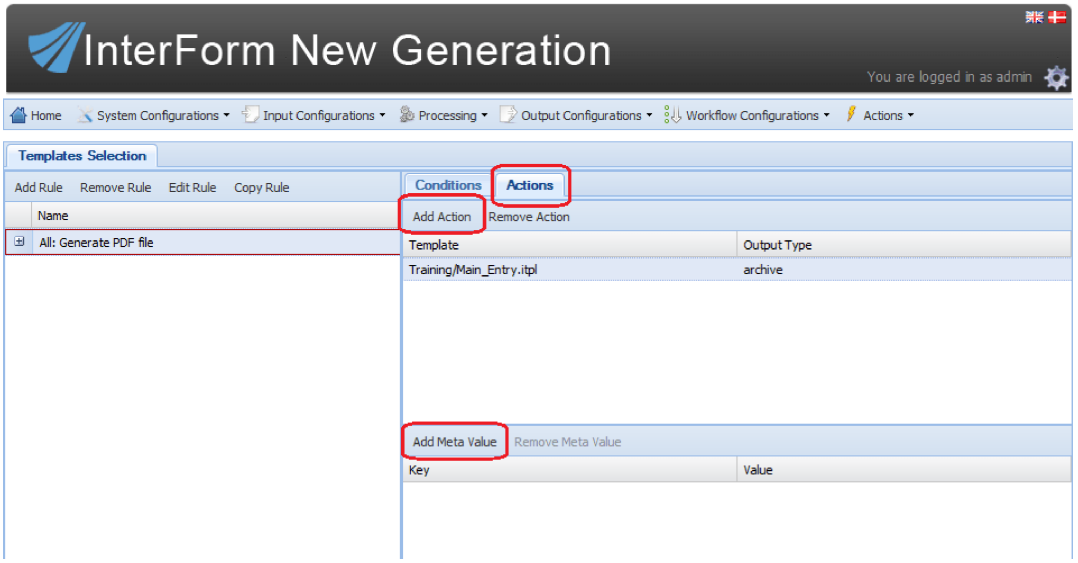
http://docs.oracle.com/javase/1.5.0/docs/api/java/util/regex/Pattern.html#sum

The expression in the screen above is a dummy condition, which is always true, so if you only want one rule, then you could start out with this. If you later want to select special XML files you should insert new rules prior the one using the dummy condition as only one rule will be selected (the first one in the list).

A blank condition (or rather a rule without a condition) will function as a dummy condition, which is always true.

Working with actions

For each rule you can setup one or more actions to be executed. Click the '**Actions**' tab to view the actions for the specific rule:



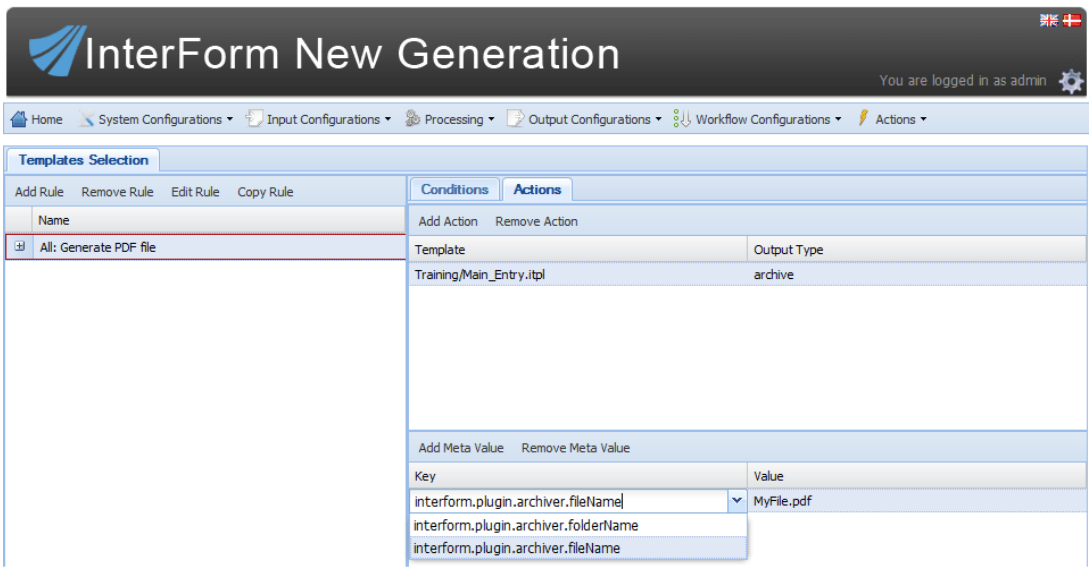
A blank action selects the normal, default action. This will mean: If the template is blank, then the default template setup in the System configuration will be used and that will decide what output to generate.

If you leave the output type and meta data blank in the screen above, then the outputType and meta data must be setup in the used template and then this will be used. If you specify anything in the screen above, then these settings will override the specifications in the template.

You can add one or more actions by clicking the 'Add Action' icon above.

Setting meta data values

For each action you can set some meta data keys if needed. To work with them you just mark an Action to view the meta data keys already set for this action. To add a new meta data key you click the **Add Meta Value** icon:



The drop down list of keys will adapt to whatever you have setup for the output type above. Above we have selected achive (for creating PDF files) thus the meta data keys listed are the ones interesting if you want to create PDF files. If you want to insert a meta data key not listed

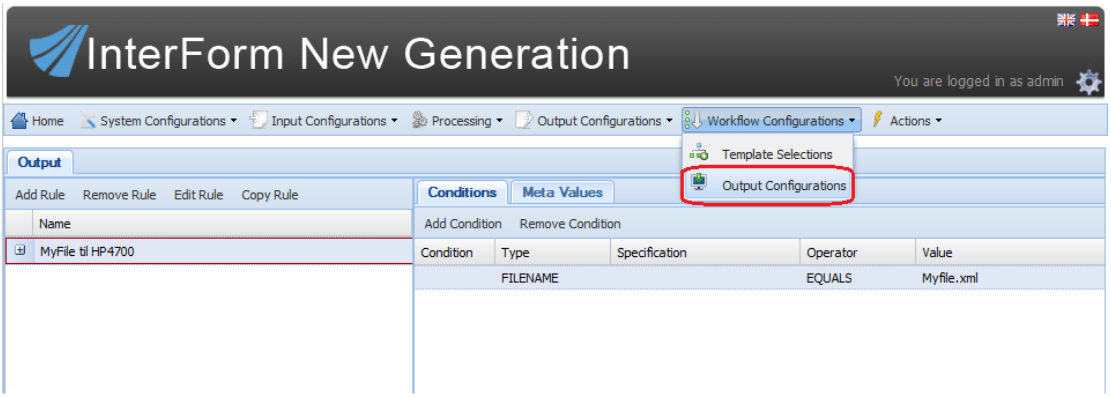
you can still just type in key. In the example above we have specified the output file name to be MyFile.pdf.

Working with Output Configurations

You can chose to redirect the printed output after the XML file has been processed. This functionality has been added to offer the maximum flexibility to those users, what are not working from a fixed location, so that they would like to select different printers depending on where they are currently working. Normally you would e.g. let the template select the output printer, but with an output configuration you can override that selection if a certain condition is met and select a different printer for those situations.

The output configurations are working independently of the template selections above, so you can here insert general overriding rules for the printed output.

You select the output configuration for the workflow from the web interface via 'Workflow Configurations' and 'Output Configurations':

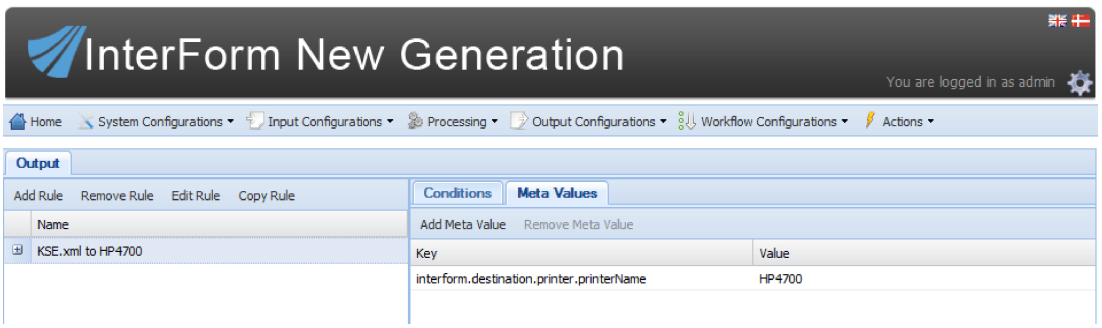


You add new Output configution rules via the 'Add Rule' on the left. The rules works as for the Template Selections:

A rule is a selection of XML files. When a new XML file arrives in the XML_inbox directory as defined in the configuration, then the Workflow (if activated) will compare the attributes and contents of the XML file with each of the rules listed until a condition for a rule is validated as true. Only the first true rule is executed, so the sequence of the rules is quite important. You can rearrange the rules simply by dragging a rule up or down in the list.

The selection of the output configurations works independently of the template configurations.

We are selecting the XML files named **Myfile.xml** above. This specific XML file is printed on a specific printer specified by setting the associated meta data value:



(We have clicked the 'Meta Values' tab to see this).

By clicking the ‘Add Meta Value’ icon you can one or more meta data values to set the output printer or other printer settings. In the case above we override the printer set in the template and set the output printer to be HP4700. (The HP4700 printer must be registered in the InterFormNG printer configuration).

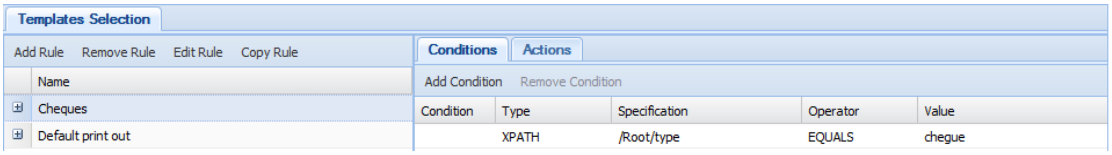
Example Rule setups

The rules that you setup for the template and output selection can be quite advanced. To help you to define the right condition a few examples are listed below:

In this case I want to print out all on a default printer - except the cheques, which are identified by the contents:

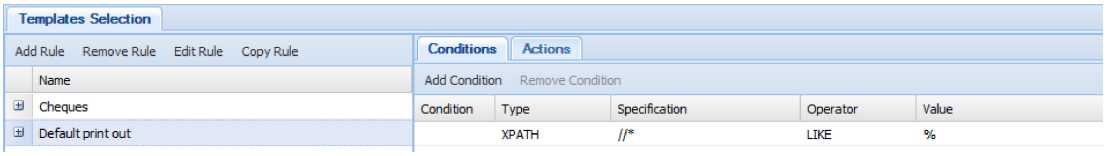
```
<?xml version="1.0" encoding="utf-8" ?>
- <Root>
    <type>cheque</type>
```

We can identify this type of XML file with this rule in the template selection:

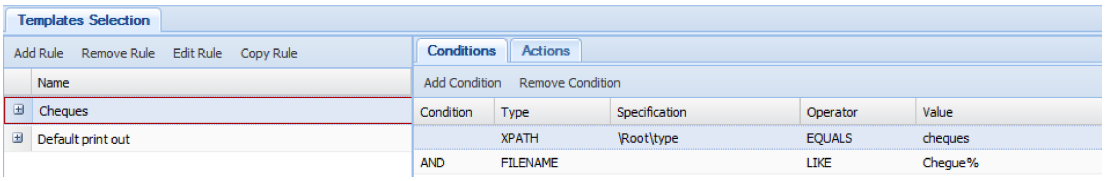


Notice, that the first rule is checking for cheques before the default one. Under Action each rule selects different printers.

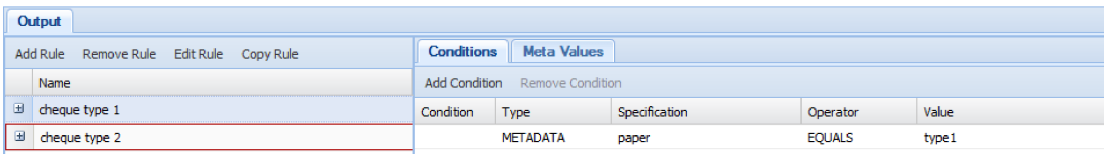
The default print out rule use a condition which is always true:



The cheques XML files might also be identified by a special file name e.g. Cheques_ followed by a date like Cheques_2013-12-12.xml. A test only valid for such file names and contents above could be:



You can even use a meta data value as a part of the condition. The meta data value could come from a database plugin - or when using the workflow output configuration it could also have been set in the template. In the template you could e.g. do some comparisons/calculations to find out what kind of paper, that should be used and signal that back to the workflow as an extra meta data:



Transforming and Splitting XML files

In some situations you might need to transform or split up the original XML file. This can e.g. be necessary if multiple documents are included in a single XML file and you want to distribute them individually as e.g. emails or separate PDF files.

The function can not only split up an XML file, but also transform it into a different structure hence this name: Transform.

An example of an XML file (example.xml) is here:

```
<orders>
    <header>InterFormNG Transform Demo</header>
    <footer>Copyright (C) 2013</footer>
    <customer>
       <customernumber mail="noreply@interform400.com">A</customernumber>
          <invoice>
             <invoicenumber>101</invoicenumber>
             <orderline>alpha</orderline>
             <orderline>bravo</orderline>
          </invoice>
          <invoice>
             <invoicenumber>102</invoicenumber>
             <orderline>charlie</orderline>
             <orderline>delta</orderline>
             <orderline>echo</orderline>
          </invoice>
    </customer>
    <customer>
       <customernumber>B</customernumber>
          <invoice>
             <invoicenumber>103</invoicenumber>
             <orderline>foxtrot</orderline>
             <orderline>golf</orderline>
             <orderline>hotel</orderline>
             <orderline>india</orderline>
          </invoice>
    </customer>
</orders>
```

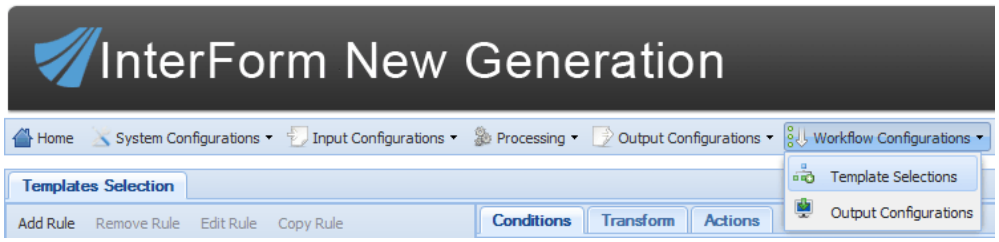
This functionality will be described while working on this xml file.

The XML file contains elements, that are in common, which might be included in each of the resulting XML files and it contains multiple nodes (Invoices) which is the basis of the split, so that each Invoice branch will result in a new XML file.

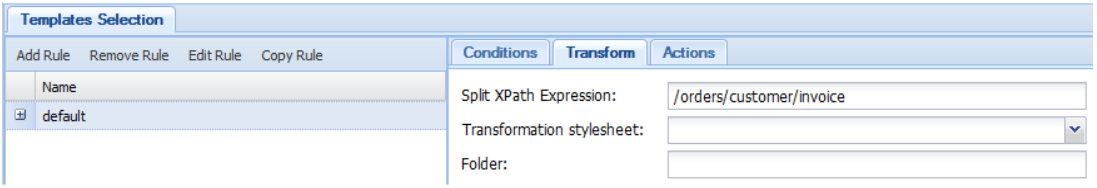
If you do not want to include any common nodes in the new xml files, then you can skip the XSLT description below and just define the path for the node, that should be the used as basis for the split.

Activating the transform function

The transform function is activated only via the workflow via the web interface, so in order to use this you first need to activate the workflow (refer to page 109 for details). Under 'Workflow' and 'Template Selections' you can setup the transform:



The next thing is to add a rule to be triggered for the xml file(s), that should be transformed. You should refer to the description of the Workflow on page 110 for details of that. When you select a rule you can now setup a transform by clicking the 'Transform' tab:



The fields are:

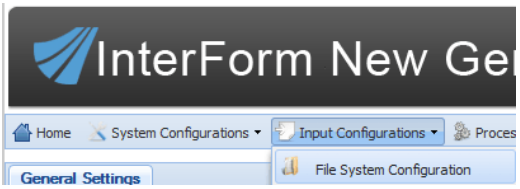
Split Xpath Expression:
The path to the detail node. If there are multiple nodes with this path, then multiple XML files will be created (split). This will later be referred to as the detail path. If you (e.g. by accident) have specified a path for the **Split Xpath Expression**, which does not exist, then the file will not be transformed. If you are afraid of typing in the wrong path (it must be 100% accurate and it is case sensitive), then you can use the graphical designer and use the text element, set it to xpath and use 'select from document' (refer to page 71) and copy if from there.

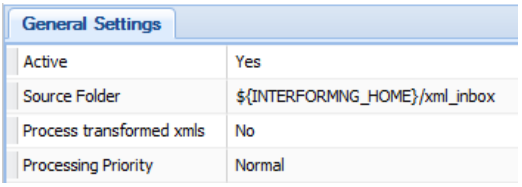
Transform stylesheet:
Here you can enter an XSLT stylesheet to setup the contents of the new XML file(s). If you keep this blank, then only the detail path will be included in the new XML file(s).

Folder:
Here you can type a path to where the transformed file(s) will be created. You can state a relative path (relative to {INTERFORMNG_HOME}) or the full path. If you keep this blank, then the XML files will be created in {INTERFORMNG_HOME}/transform_box. The use of the folder is described on page 119.

If you add any actions under the Actions tab, then these will be executed for all the new, transformed XML files.

Before we run the transformation it is a good idea to check out if the transformed/splitted XML files are used as input in the workflow. You can set that via 'Input Configurations' and 'File System Configuration':





The **Process transformed xmls** setting defines if you want to run the transformed xml files through the normal processing. This could be relevant if you want to handle each new xml file in a different manner i.e. some should emailed and others should be printed. For first time testing you could set this to 'No'. If you set this to 'Yes' and you set the output folder of the transform to the 'XML_inbox' directory, then the transformed

If you run the example.xml file as shown on page 115 through the transform definition above, then the output will be these 3 files in {INTERFORMNG_HOME}/transform_box:

```
---------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<invoice>
    <invoicenumber>101</invoicenumber>
    <orderline>alpha</orderline>
    <orderline>bravo</orderline>
</invoice>
---------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<invoice>
    <invoicenumber>102</invoicenumber>
    <orderline>charlie</orderline>
    <orderline>delta</orderline>
    <orderline>echo</orderline>
```

```
</invoice>
-----------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<invoice>
    <invoicenumber>103</invoicenumber>
    <orderline>foxtrot</orderline>
    <orderline>golf</orderline>
    <orderline>hotel</orderline>
    <orderline>india</orderline>
</invoice>
```

So by simply stating the path, that should trigger the split you will get each branch in separate files like above. So that is the simplest way to split, but the contents of the output is fixed.For more advanced splitting and transforming we will need to use XSLT, which is described on page **120**.

The files created by the transform function will (as default) be created in the {INTERFORMNG_HOME}/transform_box directory named the same as the original file with a counter added at the end e.g. Example-1.xml, Example-2.xml etc..

The files created in the tranform_box (or whatever output directory you specify) will **not** automatically be deleted, but will be overwritten if you transform an XML file with the same name again.

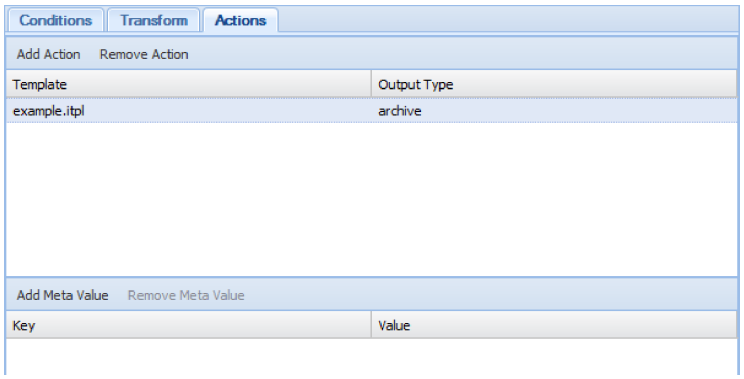
If you want to build a template for the splitted files you can e.g. copy one of them from {INTERFORMNG_HOME}/transform_box into {INTERFORMNG_HOME}/resources/document and import it in the template.

Adding actions for the transformed files

You can add actions for the transformed file(s) in one or two manners. The easiest way is to add the same actions for all the transformed files:

Same action for all transformed files

If you want to do the same action for all the files, then you can simply add the actions inside the same rule in the workflow:

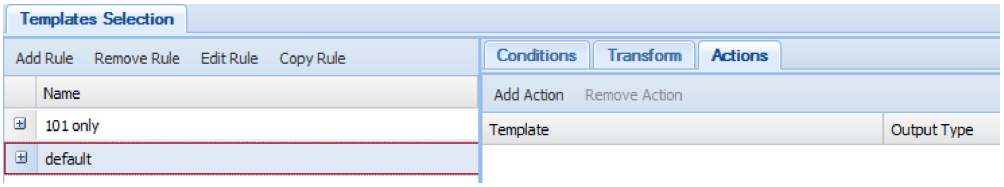


You can then let the template set the metadata to steer the actual output epending on the contents. Here we set the output pdf file name to be the the text “invoice” followed by the invoicenumber from the XML file and ending with “.pdf”:

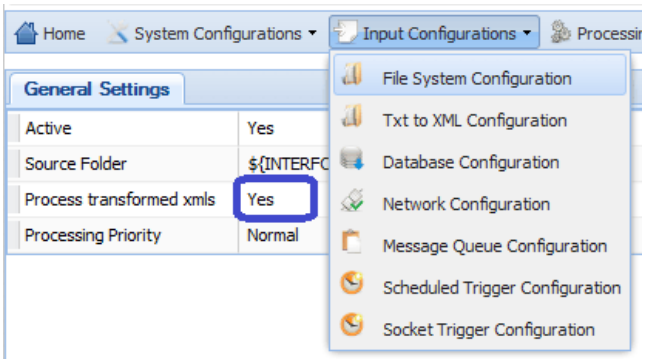
Thus creating 3 files: invoice 101.pdf, invoice102.pdf and invoice 103.pdf.

Different action for each transformed file

If you instead want to execute different actions for each splitted file, then you need to feed the transformed files back into the workflow processor. That means, that you need to remove any action for the workflow rule, which transforms:

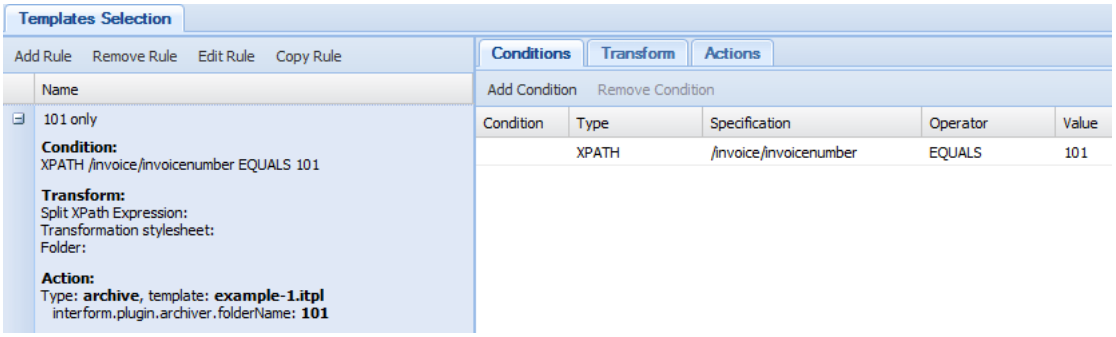


You also need to make sure, that you tell InterFormNG to process any transformed files via **Input Configurations** and **File System Configuration** and set **Process transformed xmls** to **Yes**:



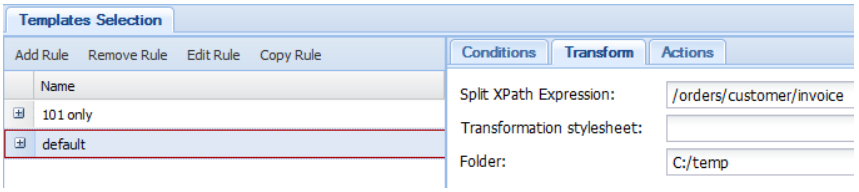
(You do not need to set the output folder for the transformed files).

Here we have defined, that only invoice 101 should be converted into a PDF file:



Creating xml files in specific folders

Please notice the folder option of the transform:



If you make sure, that the output folder for the transform is blank, then you can be sure, that your setup cannot loop where rule where the transformed files will be transformed again and so on. This is a hidden safety feature in InterFormNG: If the folder is blank, then the transformed files cannot trigger a rule using a transform definition.

However if you specify an output folder, then you can e.g. create the xml files in a folder handled by another application - or place them in the inbox directory and let the workflow decide what to do. With set setup you can transform already transformed files, but now it is your responsibility to avoid a logical loop in the setup with an infinite loop, which will just keep on creating new xml files.... (as the safety feature above is disabled when you specify a folder.)

Introducing XSLT

You can define the transformation/split via an XSLT. The stylesheet is called once for each detail branch (the exact path of the branch was defined earlier) and a detail branch (Split Xpath expression). In short the detail branch is the 'trigger' for the split - each branch will trigger a new XML file. The branch is transferred as a parameter with the name: "**detail**", so you need to define this parameter in the stylesheet:

<xsl:param name="detail"/>

In the stylesheet you can then refer to the value of this parameter as: "**$detail**". We will use this parameter to insert the detail branch when building up the structure of the new XML files.

For the example above the stylesheet could be this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
       <xsl:param name="detail"/>
       <xsl:template match="/">
               <order>
                  <xsl:copy-of select="/orders/header"/>
                  <xsl:copy-of select="/orders/footer"/>
                  <xsl:copy-of select="$detail/parent::*/customernumber"/>
                  <xsl:copy-of select="$detail"/>
               </order>
       </xsl:template>
</xsl:stylesheet>
```

The first two lines defines the stylesheet and the 3rd line defines the parameter, detail. Lines 4-11 retrieves the header, footer and customer number from the input XML file and stores them in the new xml file(s).

Next the new XML file is defined by copying the complete branches and inserting them in the structure as we want.

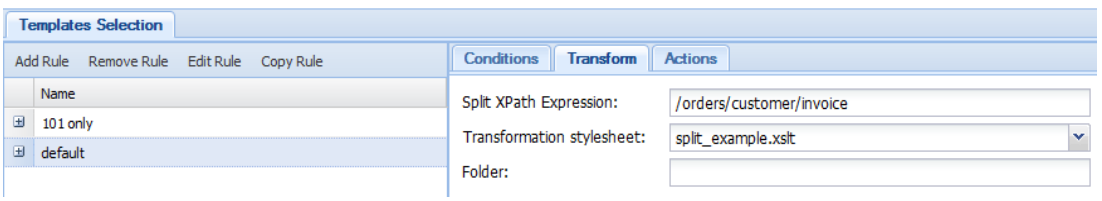
Before you use xslt you make yourself familiar with the format e.g. via online introductions like: http://www.w3schools.com/xsl/

We save the xslt as the file: **split_example.xslt** in the transform folder within the resource folder. We will use the xslt for advanced transforming below:

Advanced transforming

On page 115 we defined a simple split of the example.xml file. This resulted in 3 files, which contained the data from the detail path (Split Xpath Expression), but not the parents of this, so you could not see which customer each invoice belongs to. You can see the original xml file on page 115. Here you can see, that invoice 101 and 102 belongs to customer number A, and invoice 103 belongs to customer B.

To use an XSLT (splt_example.xlst) as defined above, we refer to this in the workflow:

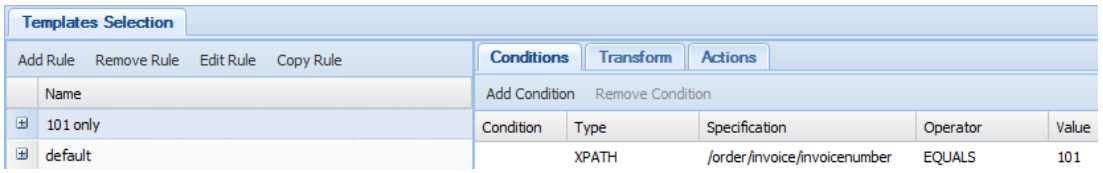


Please remember, that the paths and nodes must be written exactly as in the XML file - the Xpath expression is case sensitive.

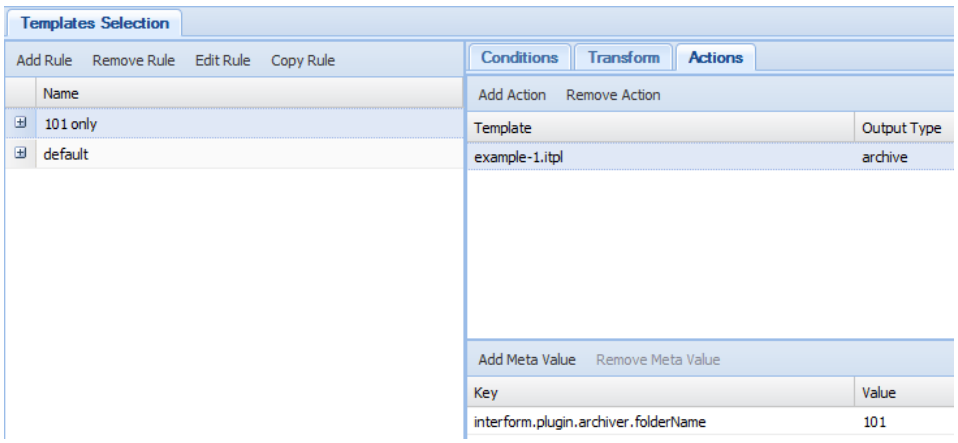
The resulting xml files looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<order>
	<header>InterFormNG Transform Demo</header>
	<footer>Copyright (C) 2013</footer>
	<customernumber mail="noreply@interform400.com">A</customernumber>
	<invoice>
		<invoicenumber>101</invoicenumber>
		<orderline>alpha</orderline>
		<orderline>bravo</orderline>
	</invoice>
</order>

<?xml version="1.0" encoding="UTF-8"?>
<order>
	<header>InterFormNG Transform Demo</header>
	<footer>Copyright (C) 2013</footer>
	<customernumber mail="noreply@interform400.com">A</customernumber>
	<invoice>
		<invoicenumber>102</invoicenumber>
		<orderline>charlie</orderline>
		<orderline>delta</orderline>
		<orderline>echo</orderline>
	</invoice>
</order>

<?xml version="1.0" encoding="UTF-8"?>
<order>
	<header>InterFormNG Transform Demo</header>
	<footer>Copyright (C) 2013</footer>
	<customernumber>B</customernumber>
	<invoice>
		<invoicenumber>103</invoicenumber>
		<orderline>foxtrot</orderline>
		<orderline>golf</orderline>
		<orderline>hotel</orderline>
		<orderline>india</orderline>
	</invoice>
</order>
```

These XML files are now being handled by the workflow again (as we told it to), but only the first XML file fits the condition in the workflow:



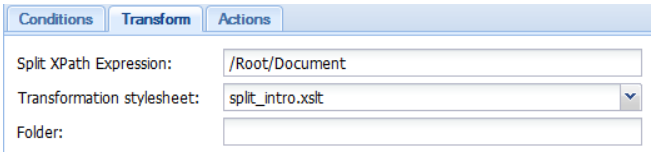
We have told it to create a pdf file in the '101' directory inside the default pdf_outbox directory:



We could have added new rules to handle the other 2 xml files, so that they could be sent as email and/or print, but that should be trivial.

Splitting the intro_demo.xml file

For the Intro_demo.xml file, which is included in InterFormNG inside {INTERFORMNG_HOME}/resources/documents/Intro you can also split up this for each Document node. This is done with this path:



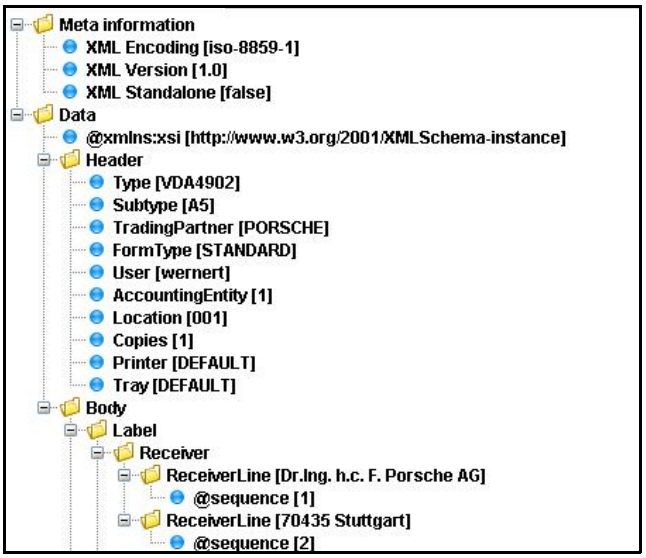
Using this xslt creates one xml file for each document - including the information from the parent node:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
	<xsl:param name="detail"/>
	<xsl:template match="/">
		<Root>
		   <xsl:copy-of select="$detail/parent::*/CompanyInfo"/>
		   <xsl:copy-of select="$detail/parent::*/Greeting"/>
		   <xsl:copy-of select="$detail/parent::*/Barcode"/>
		   <xsl:copy-of select="$detail"/>
		</Root>
	</xsl:template>
</xsl:stylesheet>
```

Appendix A: Using XPath with InterFormNG - in short

In InterFormNG you use the language XPath hen referencing to data from the XML file. There are many sources of information if you want to know more about XPath, but you can also simply read the few examples below to have a good idea of how it can be used with InterFormNG:

In XPath you can e.g. refer to data in this XML file:



If you type this in InterFormNG as the XPath:

/Data/Header/Type , then the actual text retrieved is: **VDA4902**.
Also the Xpath: */Data/Header/Type/text()* will retrieve this value.

Please note that the Xpath is case sensitive!

Starting with a '/' indicates that you are referring to the node from the root of the XML file. It the Xpath does not start with a '/', then the path is relative to a an XML file cursor. This cursor can in InterFormNG previously have been set via a Select Cursor Element in InterFormNG or via a ForEach Element:

/Data/Body/Label

If you insert the XPath above in a ForEach Element, then InterFormNG will run the loop for each label node in the XML file.

You can then inside of the ForEach loop refer to the data inside the label e.g. with an Xpath like this:

Receiver/ReceiverLine[@sequence=1]
This will retrieve the value '**Dr. Ing. H.c.F. Porsche AG**' in the label above by selecting the ReceiverLine element, where the attribute 'sequence' equals "1".

You can also retrieve an attribute. If you e.g. consider this addition to the type element above:

With this XPath specification you will retrieve the value of the attribute mytype of the Type element: */Data/Header/Type/@mytype*

So use the @-sign to refer to attributes.

If you only specify the Xpath to a level with subelements, then all subelements are also included in the result. If you e.g. state the XPath: */Data/Header* you will get this result:
VDA4902 A5 PORSCHE STANDARD wernert 1 001 1 DEFAULT DEFAULT

If you want to concatenate constants with data from the XML file you can use the **concat** function e.g. like this:

concat('ABCD',/Data/Header/Type,'EFGH')

This inserts the XML data right in between the constants: 'ABCDVDA4902 EFGH'.

You may notice, that there above is a blank between the XML data and the trailing constant. You can remove that by removing leading and trailing blanks of the XML data like so:
concat('ABCD',normalize-space(/Data/Header/Type),'EFGH')

You can also **substring** data like this: substring(/Data/Header/Type, 2,3)
The first parameter of the substring command is the string, that should be subset, the second is the start position and the third is the length, so if /Data/Header/Type contains the string 'VDA4902', the result will be 'DA4'.

Other string functions are:
substring-before(expression1 ,expression2)
substring-after(expression1 ,expression2)
replace(expression1, expression2, expression3)

substring-before searches for the text, expression2 inside expression1 and returns the part of expression1, that precedes the found position. This formular returns a 'c':
substring-before("c:\dir", ":\")

substring-after searches for the text, expression2 inside expression1 and returns the part of the expression1, that follows the found position. This returns 'dir':
substring-before("c:\dir", ":\")

replace replace any occurrence of expression 2 in expression1 with expression3. So the expression replace('abc@def','@','123') will result in the string: 'abc123def'.

translate
The function, translate is able to translate a character with another. It will search the first string for any character in the second string and if found it will use the same number in the list from the third string instead. This will e.g. replace any comma with a dot in the variable, in:

```
translate("{@in}", ",", ".")
            -S1-   S2    S3
```

If the third string is empty the character will be replaced with nothing: (commas are removed)

```
translate("{@in}", "," , "")
```

You can also convert from lower case to upper case in this way:

```
translate("{@in}", "abcde" , "ABCDE")
```

If substring-before and substring-after is not able to find the text, then an empty string is returned.

Numeric operations:
You can also do normal calculations as adding, subtracting, multiplicating, dividing in XPath. This can e.g. also be stated as an XPath expression:
*substring(/Data/Header/Type, 4,3)*5 div 6* (you need blanks around 'div')
With the value ('VDA4902') as mentioned above (for /Data/Header/Type) this will be calculated as 490 multiplied with 5 (equal 2450) divided by 6 making the result: '408.3333333333333'.

So as you can see divide is stated as '**div**' and modulo is '**mod**'.

A numeric result is converted to a string when returning the value so you can e.g. combine substring functions to the result, so we can even limit the number above to 2 decimals in this manner:
*concat(substring-before(substring(/Data/Header/Type, 4,3)*5 div 6,'.')*
*,'.',substring(substring-after(substring(/Data/Header/Type, 4,3)*5 div 6,'.'),1,2))*

Numbers must be written with a dot as decimal point without any thousand separator. To convert a decimal comma into a dot and to remove any dots (used as 1000 delimiter) you can use the **translate** function like this: **translate**(string,",",".") . Assuming you are using a dot as a decimal point and a comma as thousand separator you can remove the commas with this expression: **translate**(string,",","").

The result can then be used in numeric expressions.

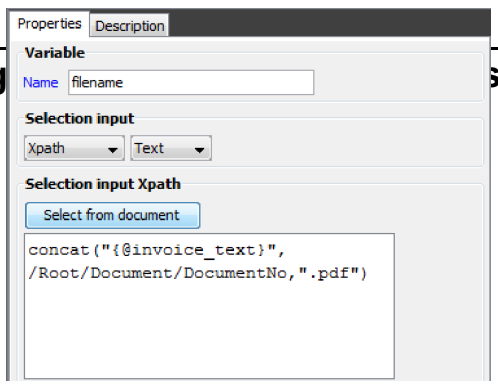
The explaination of the above is, that we concatenate the substrings consisting of:
1) The part of the amount, that is in front of the '.' i.e. '408' in this case.
2) The '.'.
3) Only the 2 first characters of the part of the amount, that is after the '.' i.e. '33' in this case.
- making the final result: '408.33'.

A more simple round of can also be done if you set the data type (in the text element) to numeric and then use this formula:
round(100(/Data/Header/Type*5 div 6)) div 100*

(The **round** function will round off to the nearest integer).

Remember that you can combine the XPath functions, and that there are many other functions not mentioned here. This appendix hopefully gives you an idea of how strong a tool this is. You can e.g. refer to this link for additional information:
http://www.w3schools.com/xpath/xpath_functions.asp

Using sions



Xpath is a strong tool in itself, but you can even use variables in your Xpath expressions. The way it works is by referring to a variable as: **@{variable}**

Here is an example:

Here the value of variable, filename is set as value of variable, invoice_text (which in this case is language dependent and contains e.g. the value “Invoice” or “Rechnung”) concatenated with the document number, which is found in the XML file and finally the extension “.pdf” is added, so that a possible value of filename could be “Invoice1001.pdf”.

Appendix B: Edit printing setup in setup files

When setting up InterFormNG for printing, the normal way would be to configure this in the web interface as described on page 25. You can however add much more functionality into the configuration files if needed. You can even write your own scripts in the configuration files to e.g. dynamically select the printer without changing the templates. Here the setup files behind the scenes are described.

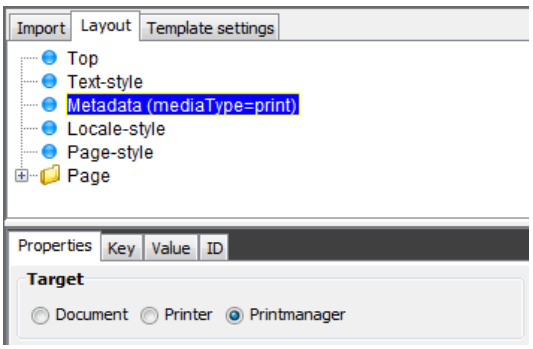
Setting up InterFormNG for printing require these steps:
(InterFormNG_Home refers to the path saved in the InterFormNG_Home environment variable)

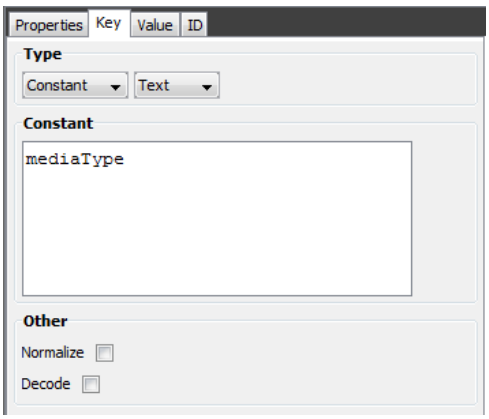
- Edit the file {InterFormNG_Home}/Profiles/default/application.properties. The file can have this content:

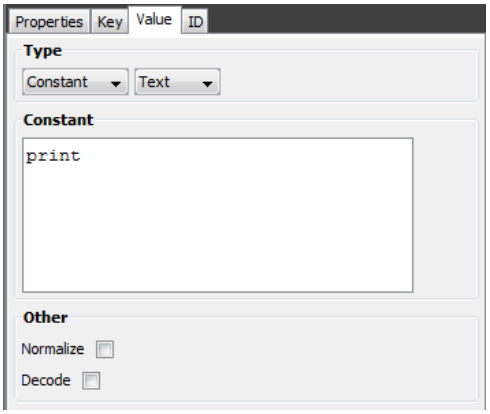
```
# Service resource settings
service.pdf.outputPath=c:/temp
service.xmlengine.defaultTemplate.filename=order_PS.itpl

# Log settings
service.log.daysToKeep=30
designer.log.daysToKeep=30

poll.enabled=true
# interval in seconds
poll.interval=5
trigger.enabled=false
trigger.port=9999
trigger.answer=Request is being processed
```

  In the file above you should select the template (design) to merge with when printing out. That is specified as '`service.xmlengine.defaultTemplate.filename`' above (So here it is set to order_PS.itpl).

- The template used should state what kind of output you want. That is done by adding this metadata in the template: (metaType=print)







- You also need to specify which printer to use for the output. This is setup in the file {InterFormNG_HOME}/basic-printmanager/conf/printConfig.xml. The structure is described below, but you can also add the necessary data in this manner: Open the command tool by selecting the <Start>, 'All Programs', 'InterFormNG' and 'CmdTool'. When selecting this option a Dos Window will open.

  If you here run the command '**Printconfig**' an example XML configuration is displayed. Now you can copy the output into the printConfig.xml file by right clicking on the head of the dos window and select 'Edit' and 'Select All'. Now right click again on the head of the window and select 'Edit' and 'Copy'. Finally paste the clipboard contents into the printConfig.xml file.

  Now edit the printConfig.xml file and change the last lines, so that you select the printer number, that you want to use - e.g. printer1:

```
<documentTypes>
    <documenttype name="*">
        <print>
            <value>true</value>
        </print>
        <printer>
            <value>printer1</value>
        </printer>
        <interactive>
            <value>false</value>
        </interactive>
    </documenttype>
</documentTypes>
```

Selecting the printer via XML

The actual printer used for printing out is selected via the basic-printmanager plugin.

The basic-printmanager plugin reads configuration data from a file called "printConfig.xml", stored in the plugin's "conf" directory (the path is defined during install as described on page 15). If this file is not available, it is created automatically by the plugin and initialized with default values that will ensure that documents are printed on the default printer connected to the current computer, and an interactive dialog similar to the print dialog found in windows applications is shown to the user just before printing the document.

Structure of printConfig.xml

The XML configuration file has the following general structure:

```
<printConfig>
    <printerTypes>
```

```
        </printerTypes>
        <printers>
        </printers>
        <documentTypes>
        </documentTypes>
</printConfig>
```

For a good example of some of the attributes available you can e.g. run **the InterFormNG Cmd application** in this way: Under 'All Programs' you select 'InterFormNG' and 'CmdTool'. When selecting this option a Dos Window will open. If you here run the command '**Printconfig**' an example XML configuration is displayed:

```
> printconfig
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PrintConfig xmlns="http://interform400.com/xml/schemas/2010/01/01/">
   <printerTypes>
      <printerType>
         <id>printer1Type</id>
         <supportedAttributes>
            <supportedAttribute name="copies"/>
            <supportedAttribute name="orientation">
               <value>portrait</value>
               <value>landscape</value>
            </supportedAttribute>
            <supportedAttribute name="mediaSize"/>
           <value>iso-A4</value>
            <supportedAttribute name="mediaTray">
               <value>manual</value>
            </supportedAttribute>
            <supportedAttribute name="chromaticity">
               <value>monochrome</value>
            </supportedAttribute>
         </supportedAttributes>
      </printerType>
   </printerTypes>
   <printers>
      <printer type="printer1Type">
         <id>printer1</id>
         <name>HP Color LaserJet 4700</name>
      </printer>
   </printers>
   <documentTypes>
      <documenttype name="Letter">
         <print>
            <value>true</value>
         </print>
         <printer>
            <value>*</value>
         </printer>
         <interactive>
            <value>true</value>
         </interactive>
         <attributes>
                  <attribute name="mediaSize">
                           <value>na-letter</value>
                  </attribute>
                  <attribute name="sides">
                           <value>two-sided-long-edge</value>
                  </attribute>
         </attributes>
      </documenttype>
      <documenttype name="A4">
         <print>
```

```
            <value>true</value>
         </print>
         <printer>
            <value>*</value>
         </printer>
         <interactive>
            <value>true</value>
         </interactive>
         <attributes>
                  <attribute name="mediaSize">
                           <value>iso-a4</value>
                  </attribute>
         </attributes>
      </documenttype>
   </documentTypes>
</PrintConfig>
```

(The actual display depends on the printers installed. The XML file has been changed.)

The first level under PrintConfig contains these elements:
PrinterTypes
Printers
DocumentTypes

Each one are described below including the attributes available within:

PrinterTypes

The printerTypes block is a list of printer types, each having a definition of which parameters are supported for that type, and which values are valid for each parameter.

Note that these values are somewhat technical and not easy to write in manually. Fortunately they may be retrieved automatically through the command line tool as shown above.

The list of supported attributes typically contain several names and value lists. As you will see later, these attributes may be set to specific values for individual documentTypes, and no value can or should be set unless the relevant printer supports that value.

Note that the “id” element contains a printer type name. This is a local name of a printer type, which you can freely define. Having these printer type definitions allows you to define a long list of identical printers that all support the same attributes by simply referencing the same printer type.

Printers

The printers block defines a list of printers, each referencing a printer type. Each entry in the list also identifies the printer network name to use.

The reason for having the printers list is so that document types can specify destination printers based on a logical id rather than the name of a specific printer. This way if a printer on the network needs to be replaced, only the name in the corresponding printer element needs to be changed – the documentType definitions can remain as they are.

DocumentTypes

At the end of the XML example above you can see how to define a document type. This block is where a document of a logical type, such as a letter or a label, is matched to a concrete printer along with a set of print attributes. An example of such a configuration might be that documents of type “Letter” should go to the HP printer on the first floor, and should be printed in color and stapled if there are more than one page. This could be done via this addition to the printConfig.xml file:

```
<documentTypes>
```

```
        <documenttype name="Letter">
            <print>
                <value>true</value>
            </print>
            <printer>
                <value>printer2</value>
            </printer>
            <interactive>
                <value>false</value>
            </interactive>
            <attributes>
                 <attribute name="Chromaticity">
                     <value>color</value>
                 </attribute>
                 <attribute name="Finishings">
                     <value>staple</value>
                 </attribute>
            </attributes>
        </documenttype>
    </documentTypes>
```

You can define several document types to cover the possible printing preferences there might be. Within each document type you can set these basic settings (as shown above):

print If **true** the PDF file will be printed. If **false** nothing will be printed.

printer What printer to use. Refer to the printer ID as defined above under 'Printers'.

interactive Should an interactive dialog be opened to ask for printing preferences? Set this to **true** or **false**.

Below is the list of printing attributes and examples of possible values. Remember that document types should only select attributes and values, that are supported by the used printer (as listed in the printer type).

Some printers might offer support for more or fewer values - check the specific printer with the printconfig command as described above.

Attribute	Description / Examples of values
Copies	An integer valued that specifies the number of copies to be printed. On many printers the supported number of collated copies will be limited by the number of physical output bins on the device, and may be different from the number of uncollated copies which can be supported.
Sides	**duplex** An alias for two-sided-long-edge one-sided Imposes each consecutive print-stream page upon the same side of consecutive media sheets. **tumble** An alias for "two sided short edge" **two-sided-long-edge** Imposes each consecutive pair of print-stream pages upon front and back sides of consecutive media sheets, such that the orientation of each pair of print-stream pages on the medium would be correct for the reader as if for binding on the long edge. **two-sided-short-edge** Imposes each consecutive pair of print-stream pages upon front and back sides of consecutive media sheets, such that the orientation of each pair of print-stream pages on the medium would be correct for the reader as if for binding on the short edge.

Finishings	Supported values include: **none** Perform no binding. **staple** Bind the document(s) with one or more staples. **cover** This value is specified when it is desired to select a non-printed (or pre-printed) cover for the document. **bind** This value indicates that a binding is to be applied to the document; the type and placement of the binding is site-defined. **saddle-stitch** Bind the document(s) with one or more staples (wire stitches) along the middle fold. **edge-stitch** Bind the document(s) with one or more staples (wire stitches) along one edge. **staple-top-left** Bind the document(s) with one or more staples in the top left corner. **staple-bottom-left** Bind the document(s) with one or more staples in the bottom left corner. **staple-top-right** Bind the document(s) with one or more staples in the top right corner. **staple-bottom-right** Bind the document(s) with one or more staples in the bottom right corner. (Continues on next page)
Attribute	**Description / Examples of values**

Finishings (continued)	**edge-stitch-left** Bind the document(s) with one or more staples (wire stitches) along the left edge. **edge-stitch-top** Bind the document(s) with one or more staples (wire stitches) along the top edge. **edge-stitch-right** Bind the document(s) with one or more staples (wire stitches) along the right edge. **edge-stitch-bottom** Bind the document(s) with one or more staples (wire stitches) along the bottom edge. **staple-dual-left** Bind the document(s) with two staples (wire stitches) along the left edge assuming a portrait document. **staple-dual-top** Bind the document(s) with two staples (wire stitches) along the top edge assuming a portrait document. **staple-dual-right** Bind the document(s) with two staples (wire stitches) along the right edge assuming a portrait document. **staple-dual-bottom** Bind the document(s) with two staples (wire stitches) along the bottom edge assuming a portrait document.
Chromaticity	An enumeration that specifies monochrome or color printing. This may be used to instruct the printer to output monochrome (grey-scale) print even if the document was originally produced in color. Supported values are: **color** Color printing **monochrome** Monochrome printing
Attribute	**Description / Examples of values**

Orientation	Supported values are: **landscape** The content will be imaged across the long edge of the medium. **portrait** The content will be imaged across the short edge of the medium. **reverse-landscape** The content will be imaged across the long edge of the medium, but in the opposite manner from landscape. **reverse-portrait** The content will be imaged across the short edge of the medium, but in the opposite manner from portrait.
MediaTray	An enumeration that specifies the media tray or bin to use for this print. The list of formally supported values is shown below. Note that the target printer may contain paper trays not listed below or offer an “Automatic” setting. Some printers support trays with custom names. You can specify any value that the printer understands. **bottom** The bottom input tray in the printer. **envelope** The envelope input tray in the printer. **large-capacity** The large capacity input tray in the printer. **main** The main input tray in the printer. **manual** The manual feed input tray in the printer. **middle** The middle input tray in the printe. **side** The side input tray. **top** The top input tray in the printer.

Attribute	Description / Examples of values
MediaSize	Specifies the media name on which to print. The list of formally supported values is shown below. Note that the target printer may support media sizes not shown in this list, and may reject some of the media sizes shown. You can specify any value that the printer understands. **a** | **b** | **c** | **d** | **e** Specify value to indicate the general paper format to use. For example, “a” for A sized paper. **executive** **folio** **invoice** **iso-a0 ... iso-a10** | **iso-b0 ... iso-b10** | **iso-c0 ... iso-c6** Standard iso sized paper, for example iso-a4 for A4 size. **iso-designated-long** **italy-envelope** | **monarch-envelope** | **na-letter** | **na-10x13-envelope** | **na-10x14-envelope** | **na-10x15-envelope** | **na-6x9-envelope** | **na-7x9-envelope | na-8x10 | na-9x11-envelope | na-9x12-envelope | na-number-10-envelope | na-number-11-envelope | na-number-12-envelope | na-number-14-envelope | na-number-9-envelope | personal-envelope** Italic and North American envelope sizes **japanese-double-postcard | japanese-postcard** Special postcard paper sizes **jis-b0 ... jis-b10** Japanese paper size variations **ledger | na-5x7 | na-legal | na-letter | tabloid** North American paper size variations **quarto**

Scripting

In the example above you will notice that values are set using the <value> element. Typically you will be setting values this way, but there is another option available that allows scripting to determine the value to use. Here is an example of how to specify the “copies” attribute using scripting:

```
<attribute name="copies">
        <expr><![CDATA[
                Calendar cal = Calendar.getInstance();
                cal.setTime(new Date());
                if (cal.get(Calendar.MONTH) == Calendar.JANUARY) {
                    return 3;
                } else {
                    return 1;
                }
        ]]></expr>
</attribute>
```

In this somewhat contrived example, the <value> element has been replaced by an <expr> element containing a snippet of script code. The logic in the code states that in january, the number of copies should be 3, otherwise it should be 1. The value of the outlying element will become the result of the return value from the script.

The scripting engine used is BeanShell, which offers a syntax that is similar to Java. For more information please refer to the official BeanShell home page, which may be found here:

www.beanshell.org

Access to meta data

In some cases, print parameters depend on meta data from the document being printed. For example, if a document is more than 4.000 pages long it should be printed in black and white rather than in color, or if a document is generated by a specific group of users or from a workstation in a specific department, it should be directed to a printer in that physical location.

To give administrators this level of print parameter control, script code is given access to meta data for the document being printed, in the form of a variable called metaData. This variable is a java Map consisting of key-value pairs with key being a String type and value being in principle any type of object, but most often another String.

One problem that might require scripting is, if a customer wants special print attributes applied to print for a specific department. An example of how this might be solved is shown here:

```
<attribute name="chromaticity">
        <expr><![CDATA[
                String departmentNumber = metaData.get(“departmentNumber”);
                if (deptNumber.equals(“4711”)) {
                    // Executive department
                    return “color”;
                } else {
                    return “monochrome”;
                }
        ]]></expr>
</attribute>
```

In the code fragment above, the department number of the document is examined and if this number equals 4711, the document is printed in color. Otherwise it is printed in monochrome.

Please note that the metaData key “departmentNumber” is used only as an example. InterformNG doesn't by itself have any special fields for department numbers, the field and its value would have to be supplied along with the document.

Some types of meta data are collected or generated during the conversion from XML to PDF, and is made available to scripts through the metaData variable. This could be the value of document variables, the total number of pages or the name of the current user.

Printing labels on Zebra printers

InterFormNG includes a plugin which is capable of rendering ZPL2 for use on Zebra printers. This allows you to use your design templates to output labels on Zebra printers.

Requirements

In order to print labels on Zebra printers you must complete the following steps:

- Connect a Zebra or ZPL2 compatible printer to the machine on which the InterFormNG service runs.
- Update the Print Configuration XML document to include a reference to the Zebra printer (Described on page 128).

- Ensure that the ZPL rendering plugin is correctly installed and configured

Connecting and installing the Zebra printer

Begin by physically connecting the printer to the network or to the local machine. Please refer to the Zebra operating manuals for more information on how to do this.

Next you must add the printer to your operating systems list of printers. For windows you can download the relevant printer drivers from the Zebra support web site. For other operating systems your options are limited to whatever support is available natively. You may use any printer driver that supports sending a raw byte stream to the device. For Linux you can use the Local Raw printer driver.

Updating the Print Configuration

Before you can print any labels you must configure InterFormNG to use the printer. This is done in the Print Configuration XML file.

First you must include a description of the printer type, as shown below:

```
...
<printerTypes>
   ...
   <printerType>
      <id>ZebraPrinter</id>
      <supportedAttributes>
         ...
      </supportedAttributes>
   </printerType>
   ...
</printerTypes>
```

In this example a printer type called ZebraPrinter is defined. The list of known attributes may be retrieved using the command line tool (as described on page 129), and is usually limited for Zebra printers. Please also note that there is currently no support in InterformNG for Zebra specific printer attributes.

Next, add a printer that references the type

```
<printers>
   ...
   <printer type="ZebraPrinter">
      <id>zebra01</id>
      <name>zm400</name>
      <inputFormat>
         <mimeType>x-application/zpl</mimeType>
      </inputFormat>
   </printer>
   ...
</printers>
```

The fragment above tells InterFormNG that a printer of type "ZebraPrinter", with the logical id "zebra01", is available with the system name "zm400". It further includes a new <inputFormat> configuration, which marks this printer as being capable of printing ZPL.

Next you must configure which document types render as ZPL rather than postscript. Once again edit the print configuration XML file as shown below:

```
<documentTypes>
   <documentType name="YourLabelDocumentType">
      <printer>
         <value>zebra01</value>
      </printer>
```

```
      <print>
         <value>true</value>
      </print>
      <attributes>
         ...
      </attributes>
   </documentType>
</documentTypes>
```

In the example above, a document type named YourLabelDocumentType is configured to be sent to the printer with the logical id zebra01. Any document which has a document type of YourLabelDocumentType will automatically be rendered as ZPL and sent to the zebra01 printer.

Installing the ZPL rendering plugin

InterformNG allows your design templates and the associated XML input data to be rendered to different formats. It is similar to the PDF rendering plugin in that it implements the RendererPlugin API interface, but it produces ZPL rather than PDF.

The ZPL rendering plugin should be installed by default by the product installer. To verify that it is correctly installed, examine the profile plugin directory for a folder called zpl-renderer. On Windows you may find it here:

C:\Users\Default\Application Data\Interform\InterformNG\profiles\default\plugins
Verify that the zpl-renderer subfolder exists, and that it contains a lib folder with several jar files inside.

ZPL rendering plugin configuration

Like other plugins, the ZPL rendering plugin reads configuration information from a file called plugin.properties stored in the conf folder:

C:\Users\Default\Application Data\Interform\InterformNG\profiles\default\plugins\zpl-renderer\conf\plugin.properties
The following configuration option is available:

zpl.targetResolution
In order to render elements at the correct sizes and positions, the plugin needs to know what the final resolution of your ZPL printer is. Typical values are:

For printers using a 6 dot/mm printhead the “dots per inch of printer” value is 152.4
For printers using a 8 dot/mm printhead the “dots per inch of printer” value is 203.2
For printers using a 12 dot/mm printhead the “dots per inch of printer” value is 304.8
For printers using a 24 dot/mm printhead the “dots per inch of printer” value is 609.6
Please refer to the documentation that came with your Zebra printer for information on what the resolution for your specific printer is.

Appendix C: Socket based XML processing

The InterFormNG service provides several ways for XML to enter the system for processing. A database may be queried periodically, a directory may be scanned and - as a recent addition, a client may connect directly to the service process and send XML over a socket for processing. This functionality is implemented by a plugin called "network-printjob".

This document provides documentation on the expected input format, the format of the processing receipt, as well as a simple example program that posts a processing request to the service and prints the results to the console.

Input format

A request to process XML is similar to a HTTP 'POST' request. It is structured as a series key/value pairs, followed by a message body of arbitrary length.

Header values are expected to be sent using a simple format resembling that of typical property files:

```
key1: value 1 (cr lf)
key2: value 2 (cr lf)
(cr lf)
```

The list of key/value headers is terminated by an empty line. After this you may send the actual XML to be processed.

Header keys

Header keys are expected to follow a naming convention in order to be recognized as InterFormNG meta values. For example:

```
x-interform-metadata-documentType: invoice
```

Notice the naming convention where a special prefix name is used to indicate that the value should be attached as meta data to the new job. Any key that begins with "x-interform-metadata" is automatically set as meta data on the resulting job. The key will be everything following the prefix, and the value will be everything following the colon. In the example above, the meta data key will be "documentType" and the value "invoice".

Result format

The result of a processing request is a sequence of key/value headers containing status information. In some cases a body containing additional information may follow the headers. For example:

```
x-interform-response: 200 (cr lf)
x-interform-response-message: Operation completed normally (cr lf)
x-interform-result-url:
file:/tmp/InterFormNG/service/127_0_0_1-01109543659659647700.pdf (cr lf)
(cr lf)
body... (optional, only used for extended error information at the moment)
```

Each key begins with "x-interform". Below is a table of the currently supported response codes:

Response key	Description
x-interform-response	A response code similar to a HTTP response code is always returned. Positive response codes are in the 2xx range, whereas error response codes are in the 5xx range. If an error code in the 5xx range is returned, a stack trace is typically returned in the response body.
x-interform-response-message	A textual message describing the operation result. If a job ended in error, the message will indicate the cause of the error.
x-interform-result-url	If a PDF was produced as a result of the processing, this header will return an URL pointing to the actual file. Note that depending on the value of the mediaType meta value, a PDF may not necessarily be available as a result of processing the XML. In this case no result url header will be returned.

Initiating a connection - A Java example

The following is a small snippet of Java code that shows how to connect to the service and provide some XML data for processing. Note that for the sake of simplicity, a lot of the typical error handling and other scaffolding has been left out.

```
 // Establish connection
System.out.println("Sending request");
Socket socket = new Socket("127.0.0.1",9091);

// Send XML request
PrintWriter pw = new PrintWriter(socket.getOutputStream());
pw.println("x-interform-metadata-documentType: invoice");
pw.println("x-interform-metadata-otherKey: other value");
pw.println("");
pw.println("<xml/>");
pw.flush();

// Tell the service that we are done. VERY IMPORTANT.
socket.shutdownOutput();

// Read results
System.out.println("Reading response");
BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
String line = br.readLine();
while (line != null) {
      System.out.println(line);
      line = br.readLine();
}
br.close();
```

Although the example is small it does illustrate a few key points. First of all it shows how to send metadata along with your request, and secondly it demonstrates how to properly close the output stream before waiting for input. It is important that you use the shutdownOutput() method on the socket, rather than calling close() on the output stream. Otherwise the socket itself will be closed and you will not be able to receive a response.

Plugin configuration

The network-printjob accepts TCP connections on a port, which by default is 9091. If you wish to change this port number, you may do so by editing the file <plugin-home-directory>/network-printjob/conf/plugin.properties for the network-printjob plugin.

For example, the following configuration will change the port number to 7777:

```
network.printjob.portNumber=7777
```

Note that the InterFormNG service must be restarted in order for this change to take effect.

A Groovy web server example

The Groovy example below shows how to define a web server to prompt for a name, insert it in an XML file, use a socket call (in InterFormNG) to convert the XML file into PDF and present the resulting PDF file in the browser:

```
-----------------------------------------------------------------------
@Grab('org.apache.camel:camel-core:2.9.0')
@Grab('org.apache.camel:camel-groovy:2.9.0')
@Grab('org.apache.camel:camel-jetty:2.9.0')

//@Grab('org.slf4j:slf4j-log4j12')

import org.apache.camel.Processor
import org.apache.camel.impl.DefaultCamelContext
import org.apache.camel.language.groovy.GroovyRouteBuilder

def WEB_PAGE = '''
   <html>
      <body>
         <form action="/" method="post">
            Please enter your name: <input type="text" name="name" />
         </form>
      </body>
   </html>
'''

def XML = '''
   <xml>
      <name>_NAME_</name>
   </xml>
'''

def TEMPLATE = 'Hello.itpl'

def generatePdf = { exchange ->
   def inputForm = new Properties()
   inputForm.load(exchange.in.body)
   def name = inputForm.name ?: 'World'

   def xml = XML.replace('_NAME_', name)

   def map = [
      'x-interform-metadata-com.interform400.xml.Template': TEMPLATE
   ]

   def props = new Properties()
   props.load(new FileInputStream(new File(System.env.INTERFORMNG_HOME,
'profiles/default/plugins/network-printjob/conf/plugin.properties')))
   if (props.enabled == 'false') throw new Exception("Network-printjob not enabled")
   def port = props.'network.printjob.portNumber'.toInteger()

   def encoding = 'UTF-8'
   def socket = new Socket('127.0.0.1', port)
   def os = socket.outputStream
   def request = new BufferedWriter(new OutputStreamWriter(os, encoding))
   map.each { key, value -> request.write "$key: $value\n" }
   request.newLine()
   request.write(xml)
   request.flush()
   socket.shutdownOutput()

   def response = new BufferedReader(new InputStreamReader(socket.inputStream,
encoding))
   props = new Properties()
```

```
    props.load(response)
    response.close()
    socket.close()

    if (props.'x-interform-response' == '200') {
        def url = props.'x-interform-result-url'?.substring(6)
        exchange.out.body = new File(url).bytes
    } else {
        exchange.out.body = props
    }
} as Processor

def context = new DefaultCamelContext()
context.addRoutes new GroovyRouteBuilder() {
    void configure() {
//      new File('_WORKSPACE').mkdirs()
//      context.properties.put('CamelJettyTempDir', '_WORKSPACE')
//      Jetty needs this

        from('jetty:http://0.0.0.0:1234/')
            .choice()
                .when { it.in.getHeader('CamelHttpMethod') == 'GET' }
                    .transform().simple(WEB_PAGE)
                .otherwise()
                    .process(generatePdf)
    }
}

println 'Starting...'
context.start()

addShutdownHook {
    context.stop()
    println 'Stopped!'
}

synchronized(this) { wait() }
```

Prerequisites:

- Groovy is installed (http://groovy.codehaus.org/)
- InterFormNG is running
- InterFormNG is accepting request from socket (Input Configurations > Network Configuration)

Run the script:

groovy WebFrontEnd.groovy

It might take some time to start the first time, as it is downloading Camel (http://camel.apache.org/).

Open a browser:
http://localhost:1234/

Enter your name and press Enter.

How it works.

Camel is started at line 69 in WebFrontEnd.groovy.
At line 75 Camel starts a Jetty server that listens on port 1234.
When a GET request is received (line 77) , it replies with the web page (line 11).
When a POST request is received (line 79), it

- extracts the form data (line 30-32)
- generates xml-data (line 34)
- opens a socket connection to InterFormNG (line 46)

- sends metadata - what template to use (line 49)
- sends the generated xml-data (line 51)
- receives the response (line 55)
- extracts the PDF-file (line 62)
- and, finally, returns the PDF-file (line 63)

Appendix D: Installing on other platforms

Installation of InterFormNG on the windows platform is described on page 12. InterFormNG can in principle be installed on any java enabled platform, but the installation procedure (and some few functions) are specific for the platform you are installing it on. Below the specifics for various platforms are mentioned.

Installing on Linux

First download the InterFormNG-shell.7z file intended for this specific version. There is no standard unzipper, but Ubuntu should find the program, p7zip itself, which you then install to unzip the file.

It is not important where the InterFormNG library is saved, but a suggestion could be /usr/local. The Shell scripts are not executables, you need to add that:

chmod u+x service-linux.sh service-web-linux.sh

Java is not standard in e.g. Ubuntu, so you need to install it yourself:

sudo apt-get install default-jre

Then you are finally ready to go.

First you start the service:

./service-linux.sh &

Then the service web is started:

./service-web-linux.sh &

Wait a few minutes and then open a browser:

http://localhost:1885/webapp-springmvc

Now you can start to configure the service as described in this manual.

When running the designer on Linux you might see a few 'issues':

When you first open a template or resource the folder might be empty. If that is the case, then just click 'cancel' and try again.

The preview might return with the error 'rundll32'

To make preview run under Linux you need to add a line to the client.properties file (inside user/profiles/default)

In the bottom of the file you will find this:

```
## External programs for displaying PDF and PCL files
PDFviewer=              rundll32 url.dll,FileProtocolHandler    ## Use default windows
PDF application
previewCommand=         /usr/bin/evince "@{absoluteFileName}"
## previewCommand.zpl=         notepad "@{absoluteFileName}"

## External programs for displaying HTML files
HTMLviewer=             rundll32 url.dll,FileProtocolHandler    ## Use default windows
HTML viewer
```

The line to add is written in bold above.

Installing on AIX

To install InterFormNG on the AIX platform you should go through the procedure below:

1) Download the InterFormNG-shell.7z file to a PC.

2) Unzip the file with 7-zip. You can download this application (freeware) from here:
http://www.7-zip.org

3) Now you need to copy the complete unzipped directory onto the AIX. For that you can e.g. FTP it with FileZilla (freeware). Filezilla can be download from here:
http://filezilla-project.org

4) Now we need to set the execute-bit for the used files. First change to the uploaded directory and the change the bit for the files: (we assume, that you have uploaded to the interformng directory, just replace this with the path you are using).

```
cd /interformng

chmod u+x *-aix.sh
```

5) Start designer (if the AIX is not headless), service and service-web with these commands: (It is assumed, that you are still in the right directory)

```
./designer-aix.sh
```

Start the service and then set the service.log file for the output log.

```
./service-aix.sh

tail -f user/profiles/default/logs/service.log
```

Start the web service and then set the service-web.log file for the output log.

After a few minutes (you can check when no more data is added to the log file) the web server should be running and you can access it via this link:
http://localhost:1885/webapp-springmvc
(Localhost should of course be replaced with the IP-address or name of the AIX server).

6) Now we can do a small test to verify that the service is running by copying a demo xml file into the default xml_inbox. With the second command we check if a PDF file has been generated - if not you should check the service.log file. The third command should created a printed output in the systems default output queue:

```
cp user/resources/document/Intro/Intro_demo.xml user/xml_inbox

ls user/pdf_outbox

cp user/resources/document/Intro/Intro_demo_print.xml user/xml_inbox
```

7) The sh scripts should be updated to set the right path for InterFormNG and the right path for Java e.g. like below: We use the vi editor to change the contents:

```
vi designer-aix.sh
```

Now change the begining of the file to this:
(/interformng should be replaced with the path to where you have placed InterFormNG. The third line could also refer to the Java_home environment variable.)

```
    cd /interformng
    export INTERFORMNG_HOME=./user
    /usr/java6/jre/bin/java …
```

```
vi service-aix.sh
```
(/interformng should be replaced with the path to where you have placed InterFormNG. The third line could also refer to the Java_home environment variable.)

```
    cd /interformng
    export INTERFORMNG_HOME=./user
    /usr/java6/jre/bin/java …
```

```
vi service-web-aix.sh
```
(/interformng should be replaced with the path to where you have placed InterFormNG. The third line could also refer to the Java_home environment variable.)

```
    cd /interformng
    export INTERFORMNG_HOME=./user
    /usr/java6/jre/bin/java …
```

Here is an example of the changed contents:

(Above the path to InterFormNG is /ng/2013.R525).

8) Now we want to install service-aix.sh and service-web-aix.sh as daemons with these commands:

```
mkitab interform:2:respawn:/interformng/service-aix.sh

mkitab interformWeb:2:respawn:/interformng/service-web-aix.sh
```

9) Fix preview in designer
You need to do the equivalent to this - dependent on the operating system:

```
vi user/profiles/default/client.properties

    previewCommand= /usr/bin/open "@{absoluteFileName}"
           # OSX

    previewCommand= /usr/bin/evince "@{absoluteFileName}"
          # Linux
```

Appendix E: Processing a Database

It is possible to let InterFormNG use database entries as input. The XML file is here to be inserted into a field. Below this possibility is described when implemented with MySQL and Linux.

InterFormNG uses JDBC and Hibernate to connect to the database.
In this example we will use MySQL, running on the same server as InterFormNG.
But any database server should work, as long as there is a suitable JDBC driver.

Using Hibernate it is possible to let InterFormNG create the needed table(s) and fields.
But you can also use an existing database.

In that case you'll have to edit job.hbm.xml to use the correct table and fields.

To access the database, InterFormNG needs a username/password and the authorization to read and update the database.

In this example we will use a database named 'interformDatabase' and a user called 'interformUser' with the password 'interformPassword'.
But you can change that, if you want to.

The Database

Make sure the database is running.
If it is not running, MySQL can be started with the following command
service mysqld start
But it should be running as a daemon.

Make sure the database and user is created and that the user has the needed authorization.
On MySQL you'll have to do something like
mysql -u root -p
create database interformDatabase;
create user 'interformUser'@'localhost' identified by 'interformPassword';
grant alter, create, delete, drop, index, insert, select, update on interformDatabase. to 'interformUser'@'localhost';*

InterFormNG don't need all the granted privileges.
alter, drop, etc. is only needed if you want InterFormNG to create the needed table(s) and fields.
InterFormNG only needs insert, select and update priviledge.
insert is only needed if you use the metadata-table (more on this later).

On a running production server it might be a good idea to only grant insert, select and update.

JDBC Driver

Make sure InterFormNG can find the JDBC-driver.
One way to do that, is to copy the JDBC-driver to Java's lib/ext/ directory.
On Fedora Linux with OpenJDK that would be something like
cp /usr/share/java/mysql-connector-java.jar /user/lib/jvm/jre-1.7.0-openjdk.x86_64/lib/ext

You can download the JDBC-driver from MySQL's homepage.

Configure InterFormNG's Database plugin

Open a browser and go to http://localhost:8080/webapp-springmvc
Select 'Input Configurations' > 'Database Configuration'

On the first tab, General Settings, enter the following

Driver Name	com.mysql.jdbc.Driver
Password	interformPassword
Database URL	jdbc:mysql://localhost:3306/interformDatabase
Username	interformUser
Active	Yes
Processing Priority	Normal

If you are using another database server, driver, database, user, etc. you'll have to change it.

On the second tab, Hibernate Settings, enter the following

Hibernate Dialect	org.hibernate.dialect.MySQL5InnoDBDialect
DDL Strategy	update
Show SQL	No

If you are using an existing database and table(s), the Hibernate Dialect might not be needed.
But if you want InterFormNG to create the database and table(s), it is needed.

BE VERY CAREFULL WITH THE DDL STRATEGY!!

On a running system it should ALWAYS be empty.
But you can use it to make InterFormNG create or update the database and table(s).
For more information, google hibernate.hbm2ddl.auto or see any book about Hibernate.

On the third tab, Hibernate Configuration, you can change the way the database table(s) is mapped to InterFormNG data.
Enter the following

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.interform400.xml.plugin.db.model">
  <class name="Job" table="JOB">
    <id name="id" column="C_PERSIST_ID">
      <generator class="native" />
    </id>
    <property name="processingStatus" column="C_WAS_PROCESSED" />
    <property name="createdDate" column="C_CREATED_DATE_TIME" />
    <property name="xmlData" column="C_XML" not-null="true"
type="com.interform400.xml.plugin.db.hibernate.BlobUserType" update="false" />
  </class>
  <query name="Job.findUnprocessedJobs"><![CDATA[
    from Job job where job.processingStatus = 0 order by createdDate, id
  ]]></query>
</hibernate-mapping>
```

Testing

If the database server is running, we are now ready to go.
Save you changes and (re)start service-linux.sh

Open a terminal and start a mysql session

```
mysql -u root -p
show databases;
    +--------------------+
    | Database           |
    +--------------------+
    | ...                |
    | interformDatabase |
    +--------------------+
use interformDatabase;
show tables;
    +-----------------------------+
    | Tables_in_interformDatabase |
    +-----------------------------+
    | job                         |
    +-----------------------------+
describe job;
    +---------------------+----------+------+-----+---------+----------------+
    | Field               | Type     | Null | Key | Default | Extra          |
    +---------------------+----------+------+-----+---------+----------------+
    | c_persist_id        | int(11)  | NO   | PRI | NULL    | auto_increment |
    | c_was_processed     | int(11)  | YES  |     | NULL    |                |
    | c_created_date_time | datetime | YES  |     | NULL    |                |
    | c_xml               | blob     | NO   |     | NULL    |                |
    +---------------------+----------+------+-----+---------+----------------+
insert into job (c_xml, c_was_processed) values ('<xml/>', 0);
select * from job;
    +--------------+-----------------+---------------------+--------+
    | c_persist_id | c_was_processed | c_created_date_time | c_xml  |
    +--------------+-----------------+---------------------+--------+
    |            1 |               1 | NULL                | <xml/> |
    +--------------+-----------------+---------------------+--------+
```

Depending on what you InterFormNG template does you might get an error.

Let's add some more information (fields) to the database.
On the tab 'Hibernate Configuration' add the following before the line '</class>'

```
<property name="referenceId" column="C_JOB_REFERENCE" />
<property name="serviceLockIdentifier" column="C_JOB_ID" />
<property name="statusCode" column="C_STATUS_CODE" />
<property name="severityCode" column="C_SEVERITY_LEVEL" />
<property name="statusMessage" column="C_STATUS_MESSAGE">
  <type name="com.interform400.xml.plugin.db.hibernate.TruncatingStringUserType">
    <param name="maxLength">1024</param>
  </type>
</property>
<property name="processedDate" column="C_PRINTED_DATE_TIME" />

<property name="processingPriority">
  <column name="C_PRIORITY" default="1" />
</property>

<map table="METADATA" name="metaData" lazy="false">
  <key column="id" />
  <index column="C_METADATA_KEY" type="string" />
  <element column="C_METADATA_VALUE" type="string" />
</map>
```

Save and restart service-linux.sh
In the mysql session you'll notice that there are now 2 tables

```
show tables;
    +------------------------------+
    | Tables_in_interformDatabase |
    +------------------------------+
    | job                          |
    | metadata                     |
    +------------------------------+
describe job;
    +---------------------+--------------+------+-----+---------+----------------+
    | Field               | Type         | Null | Key | Default | Extra          |
    +---------------------+--------------+------+-----+---------+----------------+
    | c_persist_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
    | c_was_processed     | int(11)      | YES  |     | NULL    |                |
    | c_created_date_time | datetime     | YES  |     | NULL    |                |
    | c_xml               | blob         | NO   |     | NULL    |                |
    | c_job_reference     | varchar(255) | YES  |     | NULL    |                |
    | c_job_id            | varchar(255) | YES  |     | NULL    |                |
    | c_status_code       | varchar(255) | YES  |     | NULL    |                |
    | c_severity_level    | int(11)      | YES  |     | NULL    |                |
    | c_status_message    | varchar(255) | YES  |     | NULL    |                |
    | c_printed_date_time | datetime     | YES  |     | NULL    |                |
    | c_priority          | int(11)      | YES  |     | 1       |                |
    +---------------------+--------------+------+-----+---------+----------------+
describe metadata;
    +------------------+--------------+------+-----+---------+-------+
    | Field            | Type         | Null | Key | Default | Extra |
    +------------------+--------------+------+-----+---------+-------+
    | id               | int(11)      | NO   | PRI | NULL    |       |
    | c_metadata_value | varchar(255) | YES  |     | NULL    |       |
    | c_metadata_key   | varchar(255) | NO   | PRI | NULL    |       |
    +------------------+--------------+------+-----+---------+-------+
```

Try creating a new record.
This time we also get some metadata back.

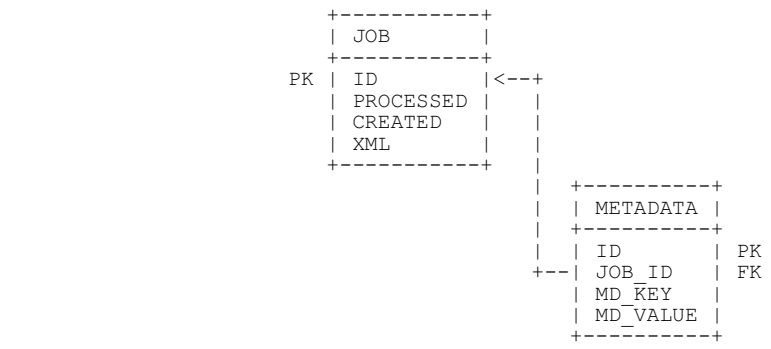
```
insert into job (c_xml, c_was_processed) values ('<xml/>', 0);
select * from metadata;
    +----+------------------------+-----------------------------------------+
    | id | c_metadata_value       | c_metadata_key                          |
    +----+------------------------+-----------------------------------------+
    |  3 | HP Color LaserJet 4700 | documentType                            |
    |  3 | media                  | interform.destination.printer.mediaSize |
    |  3 | portrait               | interform.destination.printer.orientation |
    |  3 | archive                | mediaType                               |
    +----+------------------------+-----------------------------------------+
```

This data will, of course, depend on what you do in your template.

Example: Transferring meta data values from a Database into NG

This is an example showing how you can transfer Meta data information from your database into InterFormNG.

Let's say that you want to implement the following ER-diagram:



On the SQL Server you could create the tables with the following commands:

```
SET ANSI_NULLS ON
SET QUOTED_IDENTIFIER ON
SET ANSI_PADDING ON

CREATE TABLE [dbo].[JOB](
              [ID] [int] IDENTITY(1,1) NOT NULL,
              [PROCESSED] [int] NULL,
              [CREATED] [datetime] NULL,
              [XML] [text] NOT NULL,
PRIMARY KEY CLUSTERED
(
              [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

CREATE TABLE [dbo].[METADATA](
              [ID] [int] IDENTITY(1,1) NOT NULL,
              [JOB_ID] [int] NOT NULL,
              [MD_KEY] [varchar](255) NOT NULL,
              [MD_VALUE] [varchar](255) NULL,
PRIMARY KEY CLUSTERED
(
              [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

SET ANSI_PADDING OFF

ALTER TABLE [dbo].[METADATA]  WITH CHECK ADD  CONSTRAINT [FK_JOB_ID] FOREIGN
KEY([JOB_ID])
REFERENCES [dbo].[JOB] ([ID])

ALTER TABLE [dbo].[METADATA] CHECK CONSTRAINT [FK_JOB_ID]
```

The Hibenate mapping (job.hbm.xml) could look like:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="com.interform400.xml.plugin.db.model">
              <class name="Job" table="JOB">
                            <id name="id" column="ID">
                                          <generator class="native" />
                            </id>
                            <property name="processingStatus" column="PROCESSED" />
                            <property name="createdDate" column="CREATED" />
                            <property name="xmlData" column="XML" not-null="true"
type="com.interform400.xml.plugin.db.hibernate.ClobUserType" update="false" />

                            <map table="METADATA" name="metaData" lazy="false">
                                          <key column="JOB_ID" />
                                          <index column="MD_KEY" type="string" />
                                          <element column="MD_VALUE" type="string" />
                            </map>
```

```
            </class>

            <query name="Job.findUnprocessedJobs"><![CDATA[
                        from Job job where job.processingStatus = 0 order by
createdDate, id
            ]]></query>
</hibernate-mapping>
```

Note that you'll have to do the mapping for METADATA inside the mapping for JOB.

As we don't want InterFormNG/Hibernate to mess around with the tables we created, we set hibernate.hbm2ddl.auto= in hibernate.properties, so that the hibernate.properties file looks like this:

```
hibernate.dialect=org.hibernate.dialect.SQLServerDialect
hibernate.show_sql=false
hibernate.hbm2ddl.auto=
```

Start "InterFormNG Document Processing Service".

Create a new record in the database:

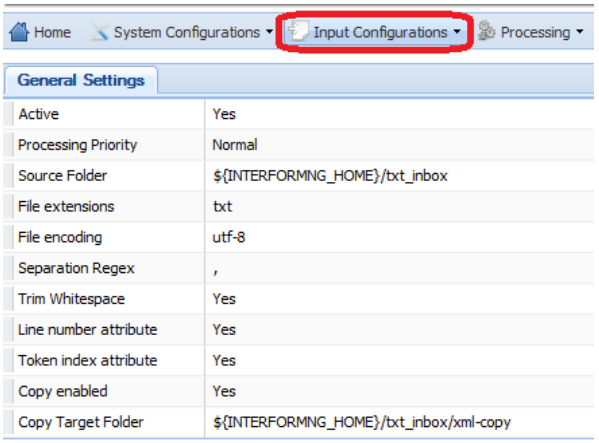
```
begin transaction;
INSERT INTO JOB (XML, PROCESSED) VALUES ('<xml><greeting>Hello</greeting></xml>', 0);
INSERT INTO METADATA (JOB_ID, MD_KEY, MD_VALUE) VALUES ((SELECT MAX(ID) FROM JOB),
'someKey', 'someValue');
commit;
```

Examples of relevant meta data keys could e.g. be the actual printer onto which the output should be printed or e.g. the pdf file name and/or path when creating PDF output or even the email receiver when sending out emails.

Appendix F: The TxtToXML Module

InterFormNG offers various ways of merging/transforming XML input, but with the TxtToXML module you can even also use text files directly as input. Currently text files can only be used as input via a monitored folder like described below.

You can configure the module via the web interface. Here you select 'Input Configurations' and 'Txt to XML Configuration':



Each parameter is explained below:

Active Select **Yes** to activate this module.

Processing Priority
The relative priority of this plugin. Select between **immediate**, **normal** and **batch**.

Source folder
The folder/directory to monitor for new text files.

File extensions
The file extensions handled by this module. Please notice, that this is case sensitive. Multiple file extensions are separated with a comma like 'txt,TXT,csv,CSV'.Files with extensions not listed are ignored.

File encoding The encoding of the input text file, use e.g. utf-8, Utf-16, iso-8859-1 etc..

Separation Regex
The character used as separator between each field/node.

Trim Whitespace
State if any leading and trailing blanks/whitespaces should be removed when converting into XML.

Line number attribute
If activated each line in the text file will be numbered in the output XML file. If activated it is possible to select a specific line from the node name.

Token index attribute
If activated each field/token/node on each lines will be numered. If not activated all fields/nodes will be named the same

Copy enabled Normally the resulting XML files goes directly into the 'engine' of InterFormNG, but if this is set the XML files will also be copied to the directory set below.

Copy Target Folder

The path of the directory where the XML files are copied to if the ‘Copy enabled’ option above is activated.

Appendix G: The PDF Security Module

If you want encrypt and password protect or digitally sign the PDF files created by InterFormNG, then you need to purchase the NG PDF Security Module. No additional installation is needed as the objects are included in the InterFormNG core.

You create the PDF files in the normal manner, and set some meta data keys for either encrypting or digitally signing the PDF files.

The settings are done by assigning values to meta data keys. Assigning meta data keys can be done in 3 ways:

1) In a template as described on page **89**.
2) In the workflow as shown on page **112**.
3) By transferring values with the XML file in a database - refer to page **152**.

Encryption and password protection

In order to encrypt and password protect the PDF files you need to assign values to these meta data keys:

Meta data Key:	**Value:**
pdfProtection.ownerPassword	<Password>
pdfProtection.userPassword	<Password>
pdfProtection.restrictions	copy, edit, print

The first two sets the password for the owner and the user. When you open the PDF file you will be prompted for a password. If you type in the owner password the PDF file will be opened with full rights.

pdfProtection.restrictions can be assigned to these values (Case sensitive):

<Blank>	The user password gives full access.
copy	Copy of text in the PDF is not allowed.
edit	Edit/changes of the PDF is not allowed.
print	Print of the PDF is not allowed.

You can combine multiple restrictions with a comma like shown above.

> **WARNING**
> *If you do not assign a value to the user password or if the user password is blank, then the PDF file will NOT be encrypted.*

Adding a digital signature

With the module it is possible to add a digital signature. You can use the digital signature to prove, who created the PDF file - in order to make it a legal document - and to prove that no one altered the document after you added your signature.

The digital signature is added as a digital certificate, that you will need to create yourself or buy from a trusted issuer.

Prerequisites

Before you can create PDF files with a digital signature you need these things:
(Apart from InterFormNG with a license, which includes the PDF Security module.)

- A **digital certificate file** (.pkcs12 or pfx), which includes the private key.
- You need retrieve the **alias** of the digital certificate - a procedure for getting that is included below.
- You need to install '**Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files**', which enables the use of 'stronger' encryption.

Below each of these prerequisites are covered:

Getting a digital certificate file

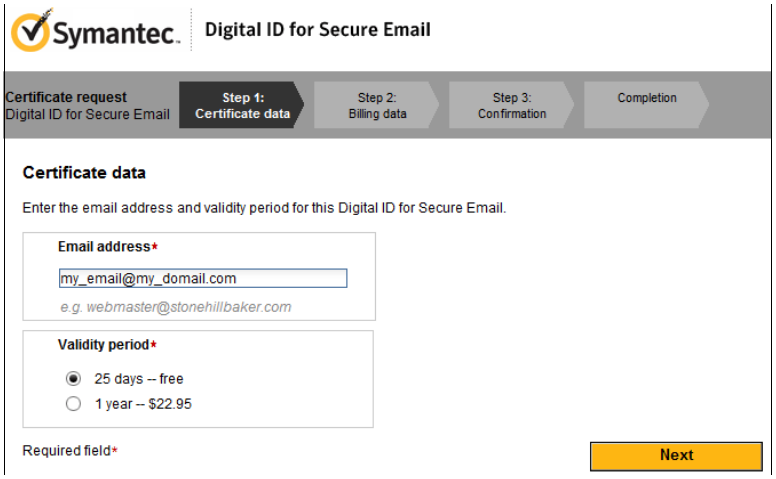
A digital pkcs12 certificate can be bought from various, trusted suppliers. As an example we will here show to get a free test certificate file from Verisign, which is valid for 25 days:

Goto this site: http://www.symantec.com/verisign/digital-id

Now click 'Buy now' on the upper right:

Symantec Digital IDs for Se...
File Edit View Favorites Tools Help
Norton Small & Medium Business Enterprise Partners
Symantec. Enterprise
United States Shopping Search
Products & Solutions Support & Communities Security Response Try & Buy
Products & Solutions / Products / Symantec User Authentication Solutions / Digital IDs for Secure Email
Symantec Digital IDs for Secure Email
Symantec Digital IDs for Secure Email allow you to digitally sign and encrypt your digital communications using a
NEXT STEPS
Buy Online

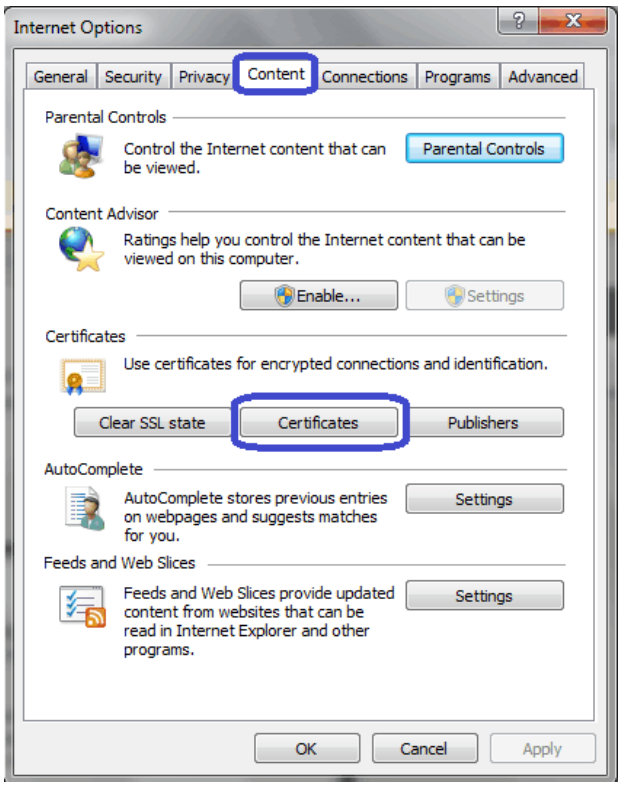
You now type in your email address and select '25 days - free' - unless you want to buy:



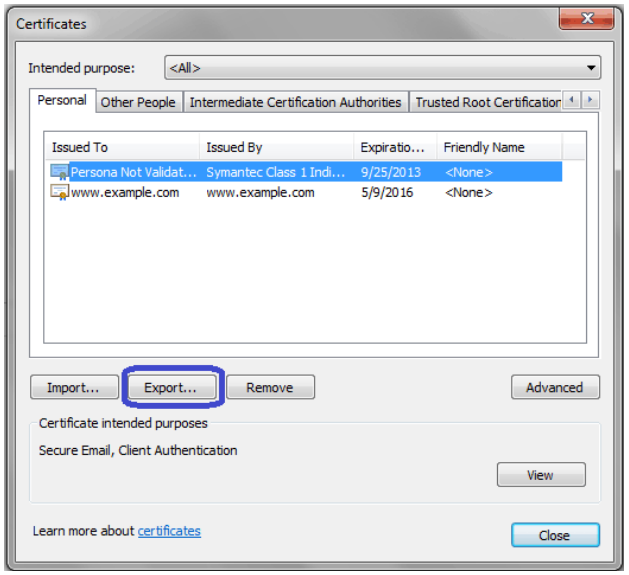
- and click 'Next'.

After accepting conditions you will now receive 2 emails: One with a link and another with a password. After signing on you can install the certificate in your browser - e.g. MS Internet Explorer.

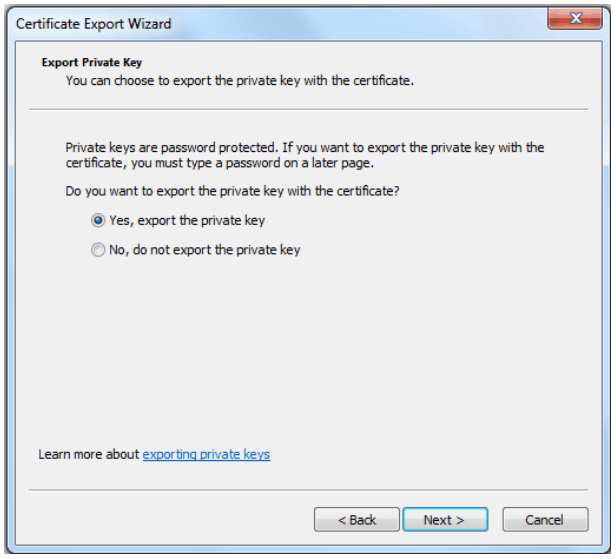
From the browser you now need to export the certificate as a pkcs12 file - including the private key. That is in MS Internet Explorer done by selecting ‘Tools’, ‘Internet options’ and the ‘Content’ tab - and finally the ‘Certificates’ icon:



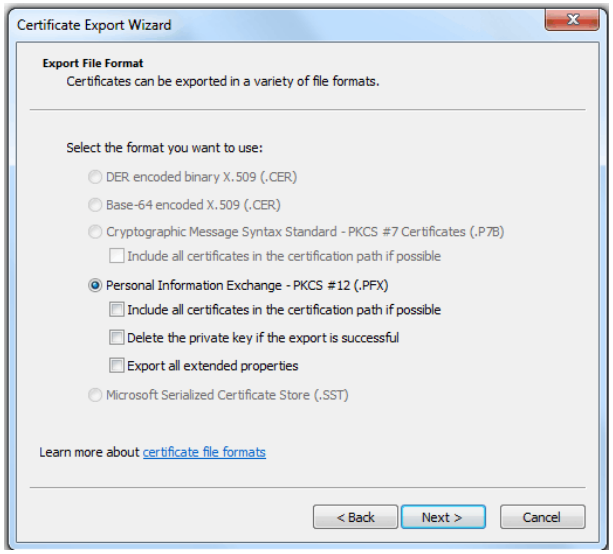
Now mark the certificate in question and click ‘Export..’:



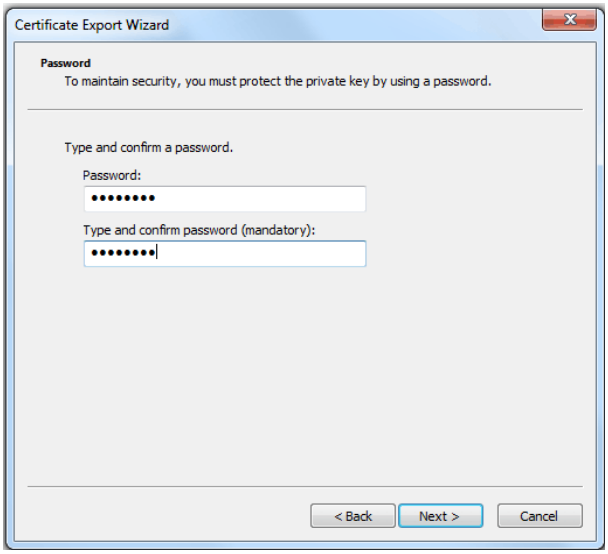
Now remember to export the private key:


Certificate Export Wizard
Export Private Key
You can choose to export the private key with the certificate.
Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.
Do you want to export the private key with the certificate?
Yes, export the private key
No, do not export the private key
Learn more about exporting private keys
< Back
Next >
Cancel

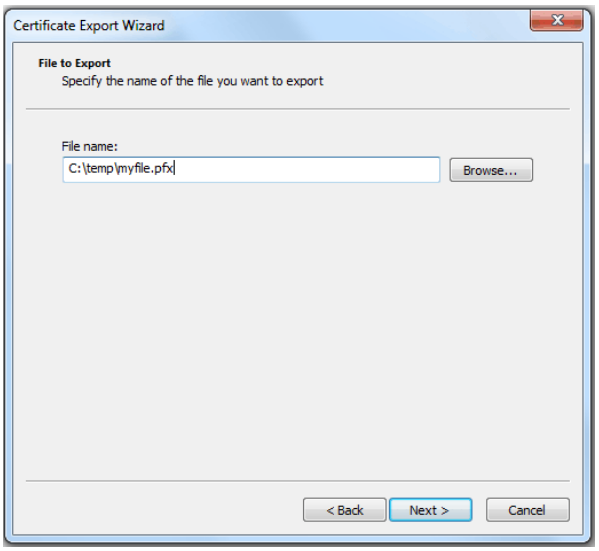
Do not activate any of the extra settings here:


Certificate Export Wizard
Export File Format
Certificates can be exported in a variety of file formats.
Select the format you want to use:
DER encoded binary X.509 (.CER)
Base-64 encoded X.509 (.CER)
Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
Include all certificates in the certification path if possible
Personal Information Exchange - PKCS #12 (.PFX)
Include all certificates in the certification path if possible
Delete the private key if the export is successful
Export all extended properties
Microsoft Serialized Certificate Store (.SST)
Learn more about certificate file formats
< Back
Next >
Cancel

It is highly recommended to protect the certificate file with a password:


Certificate Export Wizard
Password
To maintain security, you must protect the private key by using a password.
Type and confirm a password.
Password:
Type and confirm password (mandatory):
< Back
Next >
Cancel

Finally we specify the path and file name:



You can place the certificate file anywhere and later refer to the complete path or you can place it is {INTERFORMNG_HOME}/resources/certificates and then you only need to refer to the file name and subdirectories - if used.

Retrieve the alias from the certificate file

You can find the alias from a certificate file by opening a DOS session and running this command:

keytool -list -keystore "C:\temp\myfile.pfx" -storetype pkcs12 -storepass password

Inside the "" after 'keystore' you write the file (including the path), that contains the certificate. (Above the file is 'C:\temp\myfile.pfx') and the final word is the password used for locking down the certificate file (above it is 'password').

The command returns this:

```
C:\Users\kse>keytool -list -keystore "C:\temp\Kim_pw=mypw.pfx" -storetype pkcs12
 -storepass mypw

Keystore type: PKCS12
Keystore provider: SunJSSE

Your keystore contains 1 entry

le-7e788873-11e0-4384-9f8b-38ac1ced5d1b, Aug 30, 2013, PrivateKeyEntry,
Certificate fingerprint (MD5): 51:FC:D5:33:09:04:B4:C2:A9:27:F5:85:95:DF:9A:4D
```

The alias is what is preceding the first comma. In this example the alias is: 'le-7e788873-11e0-4384-9f8b-38ac1ced5d1b'. You need to save your alias as this is needed when you want to digitally sign a PDF file.

Install Java Cryptography Extension (JCE)

Additional software is needed to do the strong encryption, which is used. You will need to download and install this into the existing java installation. The complete name of the software is called: '**Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files**'.

The software to download depends on which version of JRE you are running. If you are running **JRE 7**, then you can currently use this link:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Where it is found in the bottom. After downloading you should read the 'README' file for instruction of how to install, but it is actually just a question of copying two .jar files into the /lib/security directory where java is installed. This could be here:
C:\Program Files\Java\jre7\lib\security

Inside the 'security' directory you may find two existing files named the same as the files you want to copy. If you want to be sure you can e.g. rename the existing files - then you can always come back if something should go wrong.

If you are running JRE 6 you can currently download Java Cryptography Extension (JCE) via this link:

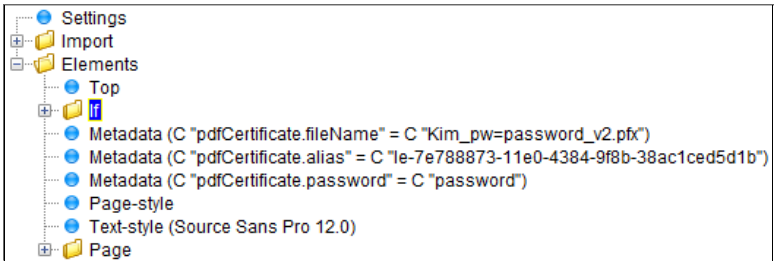
http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-plat-419418.html#jce_policy-6-oth-JPR

Signing a PDF file

If you have fixed the prerequisites you can now start to digitally sign PDF files. This is simply done by assigning values to these meta data keys:

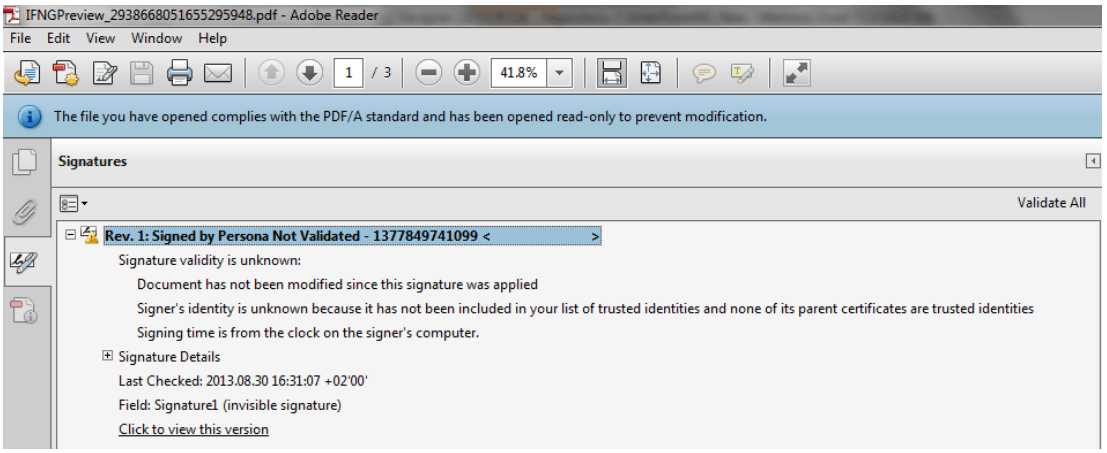
Meta data key:	**Description:**
pdfCertificate.fileName	The certificate file. Should contain the absolute path or the path relative to {INTERFORMNG_HOME}\resources\certificates.
pdfCertificate.alias	A pkcs#12 is able to contain multiple certificates. So you need to state the alias of the one to use.
pdfCertificate.password	The password needed to open certificate file.

Here is an example of how this can be setup:



Above the certificate file is placed in {INTERFORMNG_HOME}\resources\certificates, has an alias that is 'le-7e788873-11e0-4384-9f8b-38ac1ced5d1b' and the password for the file is simply 'password'.

The signed PDF file may now look something like this:


IFNGPreview_2938668051655295948.pdf - Adobe Reader
File Edit View Window Help
1 / 3
41.8%
The file you have opened complies with the PDF/A standard and has been opened read-only to prevent modification.
Signatures
Validate All
Rev. 1: Signed by Persona Not Validated - 1377849741099 <
Signature validity is unknown:
Document has not been modified since this signature was applied
Signer's identity is unknown because it has not been included in your list of trusted identities and none of its parent certificates are trusted identities
Signing time is from the clock on the signer's computer.
Signature Details
Last Checked: 2013.08.30 16:31:07 +02'00'
Field: Signature1 (invisible signature)
Click to view this version

You might think this indicates a problem, but it only states, that the signature is trusted (and the issuer is also not trusted), so you will need to trust the issuer or the current certificate.

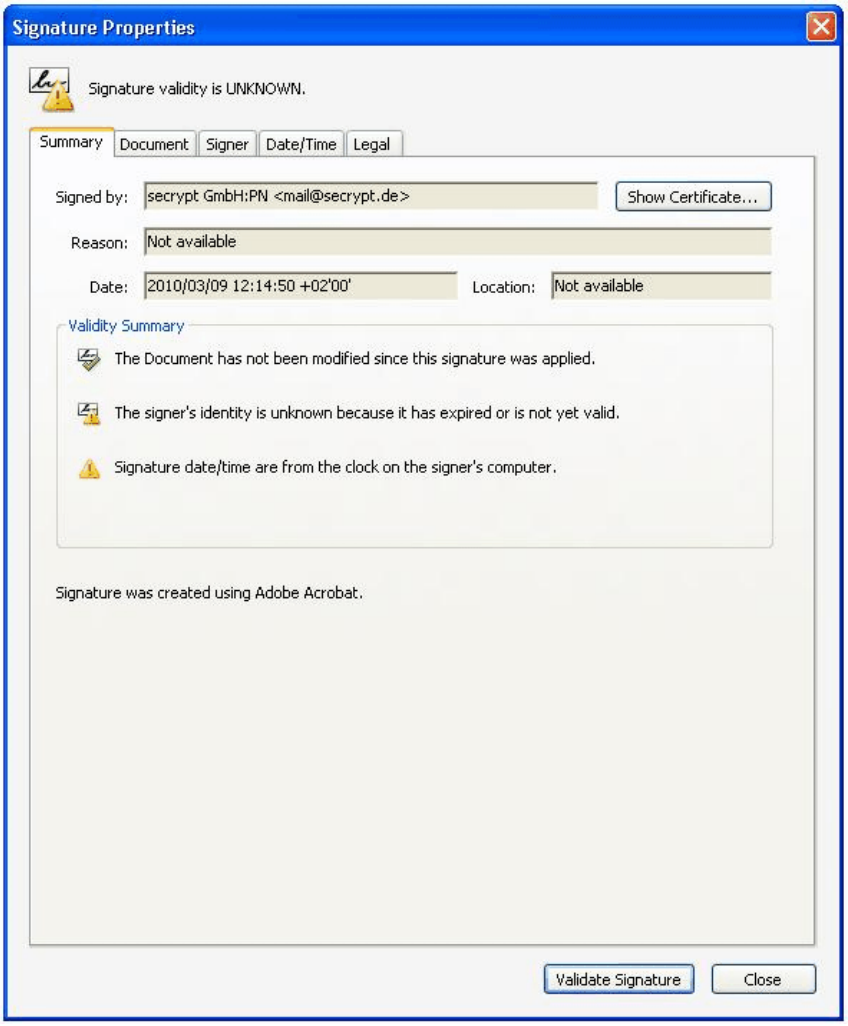
Trusting signatures

When you receive a PDF file with a digital signature you need to decide if the signature can be trusted or not. If the signature (and its parent certificate is not trusted either) you will see something simular to this under 'Signatures' in Acrobat Reader 5.0 or later:

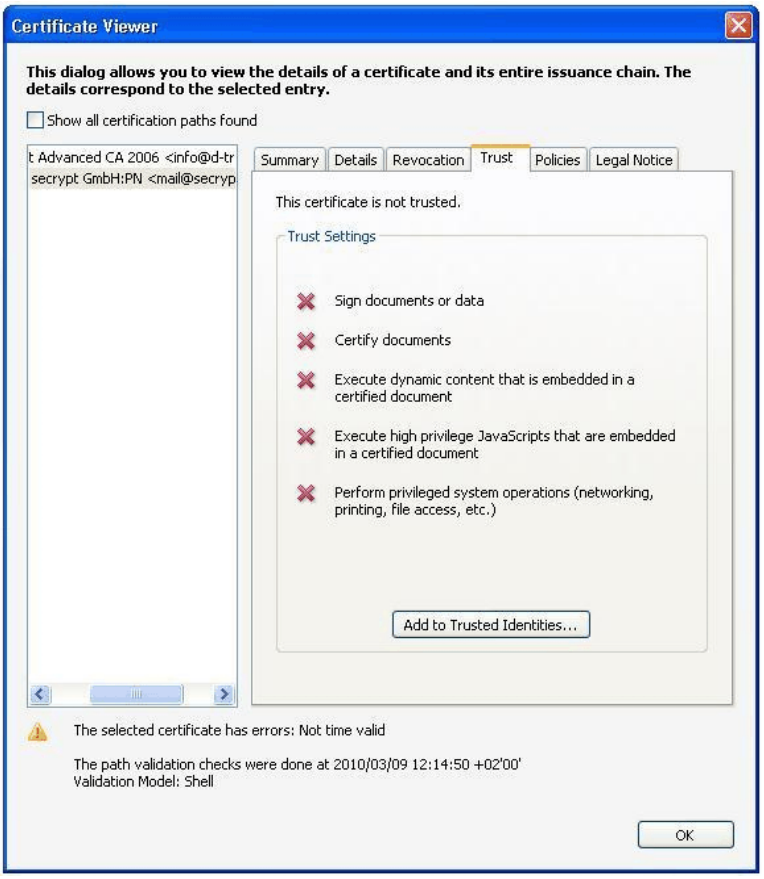
Recipient Signatures
The following people have digitally signed this document
Signed by XXXXX YYYYYY
Signature validity is unknown:
Document has not been modified since the signature was applied
Signers identity is unknown because it has not been included in your list of Trusted Identities and none of itss parent Certificates are Trusted Identities
.....

Now you would probably want to trust this identity. You can do this in these manners:

1) Right click on the 'Signed by XXXX' text above. Click 'Properties..' and you will see this:



Then select '**Show certificate**' and then another window will open (where you click the 'Trust' tab:


Certificate Viewer
This dialog allows you to view the details of a certificate and its entire issuance chain. The details correspond to the selected entry.
Show all certification paths found
t Advanced CA 2006 <info@d-tr
secrypt GmbH:PN <mail@secryp
Summary
Details
Revocation
Trust
Policies
Legal Notice
This certificate is not trusted.
Trust Settings
Sign documents or data
Certify documents
Execute dynamic content that is embedded in a certified document
Execute high privilege JavaScripts that are embedded in a certified document
Perform privileged system operations (networking, printing, file access, etc.)
Add to Trusted Identities...
The selected certificate has errors: Not time valid
The path validation checks were done at 2010/03/09 12:14:50 +02'00'
Validation Model: Shell
OK

If you here click 'Add to Trusted Identities..' you can decide to trust the included signature in Acrobat Reader'. Note, that you will get a warning telling you that this is usually unwise as you then need to be 100% sure, where this document comes from...

2) Another way to trust a digital signature in Acrobat Reader is to select 'Document' and 'Trusted Identities' in Acrobat Reader. Then click 'Add Contacts..' and then 'Import' or 'Browse' to find a file with the public key of the digital signature, that you want to trust.

3) The third way to trust certificates in Acrobat Reader is to trust those, that has been trusted in Windows. You can set that up by selecting 'Edit', 'Preferences', 'Security', 'Advanced preferences' and 'Windows integration'.

Verifying Digital Signatures

Some digital signatures are sometimes stolen or misused. To handle that situation each issuer of digital signatures lists a number of digital signatures that are revoked (Revocation lists - RCL's). So it is a good idea to check with the issuer that the current signature still is valid. You can verify the signature in at least two manners:

1) Verify during open:
You can e.g. setup Acrobat Reader to verify digital signatures when opening a PDF file by selecting 'Edit', 'Preferences' and 'Security'. Here you will see an option 'Verify signatures when the document is opened'. If you activate this option Acrobat will check the signatures when opening the PDF file. The downside is, that opening takes a bit longer.

2) Verify the signature manually:
Find the signature of the PDF document and right click on the '**Signed by XXXX**'. Then select 'Validate Signature'.

Appendix H: Conversion of pre 2014 Templates

When **upgrading** from a **pre 2014** version to a 2014 version (or newer) the templates will be converted into a new format. This has only importance if you are using barcode styles. If you are not using barcode styles you will not see any difference in the output and you can also open a template in any version of the designer no matter what version that created the template. If you are using barcode styles you cannot directly open a converted template in a pre 2014 version as the barcode styles are stored in a different manner.

The changes to the barcode style are in short:

- The barcode style MUST have a **specific barcode type**, where you could previously use the 'Unset' value as well to set a general value for whatever barcode type you use.
- **Min.** and **max.** values are converted into **one value** for the **module width** and **module ratio**. The new values represents the same outcome as the original set of min. and max. Barcode styles where the type is 'Unset' are ignored in the 2014 versions until you set a specific barcode type.
- The ratio was **previously specified in 1/1000** (or promille). During conversion it is converted into the **'real' ratio** between the wide and narrow lines (typically a value **around 2 to 3**). In the 2014 versions you specify this real ratio.
- The **properties** of a barcode style now **adapts** to the **specific barcode type** you select. Previously all properties where displayed for all types. This change makes it easier to define barcodes as only the relevant properties are shown.
- The **error correction for QR and PDF417** barcodes was previously not implemented and thus ignored. Now the value is used and may result in a different size of the 2D barcode in PDF, PCL and postscript.
- The resulting **size of the barcode may change** slightly when you compare the postscript/PCL output with the result prior to the upgrade.
- As **checksum** you could previously select the value, 'Auto'. This value was not fully implemented and the support for this value has been removed. Any template using 'Auto' will fail with an error in the 2014 versions.

The barcode style is described in dept on page 64.

Testing templates when upgrading to a 2014 Version or newer

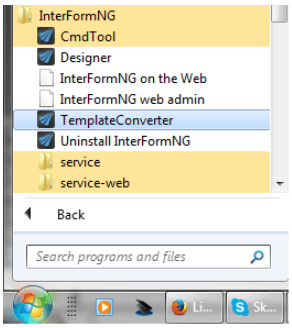
After an upgrade of InterFormNG it is recommended to do test print outs to verify that the barcodes has been converted OK. If you have any concerns about this conversion: Please contact your local InterForm support. Feel free to contact Us also prior to the upgrade if you want a free analysis of what changes you may expect if/when upgrading.

Conversion tool

When you do an upgrade from a pre 2014 version to a 2014 version or newer, then all templates need to be converted. The most efficient way to do that is to use the conversion tool, which you asked to start as the last step of the upgrade.

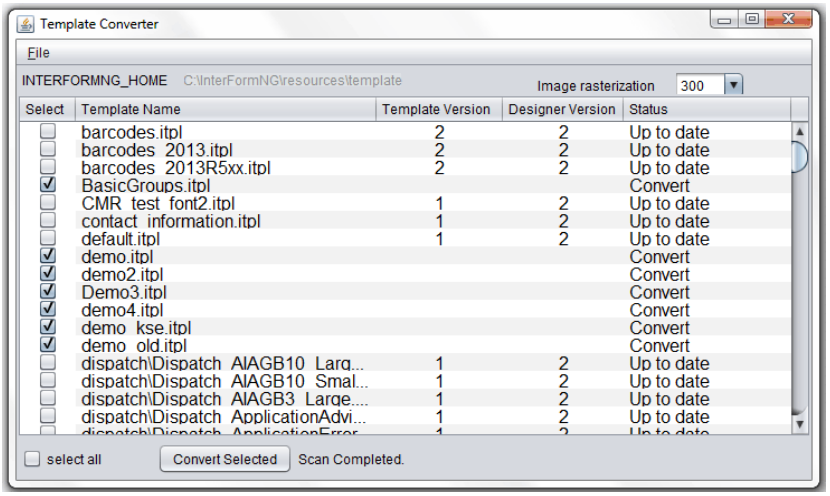
You can also chose to only convert some of your templates at a time and run the conversion tool by selecting ‘Start’, ‘Programs’, ‘InterFormNG’ and ‘TemplateConverter’:

You can also find the Conversion tool here:
C:\Program Files\InterFormNG\TemplateConverter.exe



It is recommended to convert the templates. If you do not, then the templates will be converted each time the template is opened in either the designer or the processor service causing a slower performance. Saving an open template from the designer will save it as a converted template - but not any referenced templates.

When you start the conversion tool you will see this:



The normal directory (and subdirectories) containing the templates is used as default, but you can open another directory via **File** and **Open** if you want to convert templates outside this directory.

The conversion tool first scans all the templates and lists them like above.

These are the columns:

Select A click box, where you can select/deselect the templates. As default any unconverted templates are selected.

Template Name
The relative path and file name of the template(s).

Template Version
The version of the template. This indicates if the converted template can be used in pre 2014 versions or not and also if any barcodes are used. These are the values:

1: The template can be used by any version of InterFormNG.
2: The template can only be used by 2014 versions and newer.
Blank: The template has not been converted.

Designer Version

This is the indicator if a template has been converted and if so: To which level it has been converted.

Blank: The template has not been converted.

2: The template has been converted into level 2 - it is compliant with the 2014 versions.

Status The status of each template. The status can be:

Up to date	The template has previously been converted OK.
Convert	The template has not been converted yet.
Converted	The template has just been converted.
Error...	Potential error during conversion.

> **WARNING**
>
> *If you open a template in an older version of the designer and the template was created/edited in an newer version of the designer and the template is using newer functionality not implemented in the older version, then any references to the newer functionality may forgotten when the template is opened and saved in the old designer.*

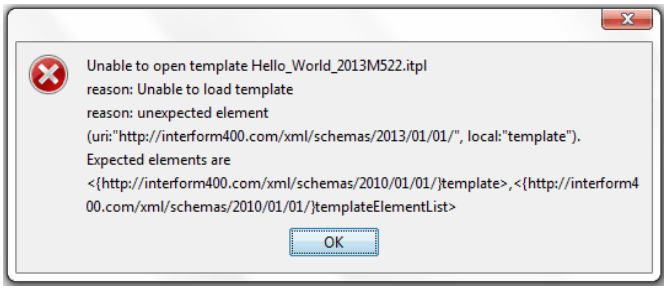
Click the **Convert Selected** icon to start the conversion.

The original unconverted templates are stored in a zip file in {INTERFORMNG_HOME}\resources\template named Convert_yyyyMMdd_HHmmss.zip

Problem when opening templates in pre 2014 Versions

If you try to open a template in a pre 2014 version, which has been converted in a 2014 version or newer, then you will get this error message if it contains a barcode style:

(The schema http://interform400.com/xml/schemas/2013/01/01 indicates the new layout)



This indicates, that the template was created for a newer version and that the barcode styles are not supported in the current (pre 2014) version.

Using unconverted templates

Each time you open a template created in a pre 2014 version, the template will be converted into the 2014 format. If you open and save a template in the designer then it will be saved in the converted format. (Any barcode style in the template will prevent it from being opened/used in a pre 2014 version). Please notice that any imported/referenced template are not updated and saved when you open the main template - they are only temporarily converted for the current 'job'.

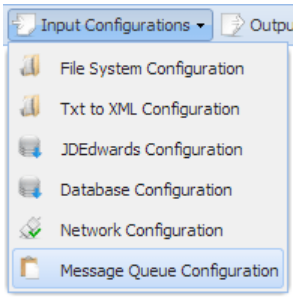
Any referenced template (also used by the processor service) will temporarily be converted at runtime each time when used. So until you have converted all your templates you may experience lower performance in the designer and the processor service.

Appendix I: Using a Message Queue for input

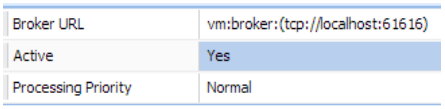
One way to get XML files into InterFormNG is to let the application monitor a message queue for entries. Below it is described how you can set that up:

Enabling message queue in InterFormNG

First you need to tell InterForm400 to start to monitor a message queue. To do that you open the InterFormNG web interface and select Input Configurations > Message Queue Configuration:



Change Active to Yes:



Now save the change and restart the processor service as described on page 100.

InterFormNG is now using an embedded MessageMQ. It is possible for other programs to access it on port 61616.

External message queue

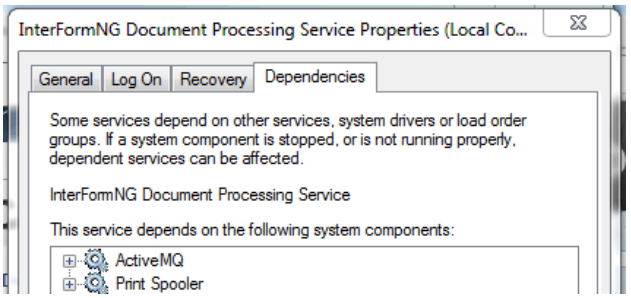
If you are using an external ActiveMQ, you'll have to change the Broker URL.
The default for ActiveMQ is `tcp://localhost:61616`.

When using an external message queue it might be a good idea to let the InterFormNG Document Processing Service depend on it.

Execute the following command as Administrator

```
sc config InterFormNGProcessor depend= Spooler/ActiveMQ
```

Now InterFormNG service is dependent on ActiveMQ


InterFormNG Document Processing Service Properties (Local Co...
General
Log On
Recovery
Dependencies
Some services depend on other services, system drivers or load order groups. If a system component is stopped, or is not running properly, dependent services can be affected.
InterFormNG Document Processing Service
This service depends on the following system components:
ActiveMQ
Print Spooler

Now starting the InterFormNG service, will also start ActiveMQ and stopping ActiveMQ will also stop the InterFormNG service.

Sending data to message queue

Using a message queue with InterFormNG is standard JMS.
It doesn't matter if you use plain Java or a framework like Spring or Camel.
For simplicity I'll use Camel.

InterFormNG will use the queue `interform.xmlProcessingRequestQueue` and the JMS message type `javax.jms.BytesMessage`.

For InterFormNG an XML-file coming via a message queue is no different form an XML coming from the filesystem.
If workflow is disabled, the default template will be used (System Configurations > General System Settings).
If workflow is enabled, the first rule where the conditions matches will be used (Workflow Configurations > Template Selections).

The simplest possible scenarie is to send the XML-data only.
But you can add headers to control the processing.
All InterFormNG metadata is prefixed with the string "x-interform-metadata-" and to make Camel unescape the "-" I've added jmsKeyFormatStrategy=passthrough to the endpoint.

Pom.xml:

```
<?xml version="1.0"?>

-<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
       http://maven.apache.org/xsd/maven-4.0.0.xsd"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns="http://maven.apache.org/POM/4.0.0">
       <modelVersion>4.0.0</modelVersion>
       <groupId>com.interforn400</groupId>
       <artifactId>messagequeue</artifactId>
       <version>0.0.1-SNAPSHOT</version>
               -<dependencies>
                  -<dependency>
                       <groupId>org.apache.activemq</groupId>
                       <artifactId>activemq-camel</artifactId>
                       <version>5.9.0</version>
                  </dependency>
               </dependencies>
</project>
```

x-interform-metadata-mediaType	archive, email or print
x-interform-metadata-documentType	all the usual metadata for archive, email and print
x-interform-metadata-com.interform400.xml.Template	which template to use
x-interform-metadata-xmlFromSocket	skip final step - only create artifact, but don't save, mail or print
x-interform-metadata-interform.printjob.processingpriority	0=Immediate, 1=Normal, 2=Batch
x-interform-metadata-interform.plugin.integration.folderName	where to store temporary files

The MQDemo.java source:

```
package com.interform400.messagequeue;

import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import org.apache.activemq.ActiveMQConnectionFactory;
import org.apache.camel.CamelContext;
import org.apache.camel.ProducerTemplate;
import org.apache.camel.component.jms.JmsComponent;
import org.apache.camel.impl.DefaultCamelContext;
```

```
public class MQDemo {

	public static void main(String[] args) throws Exception {
		String url = "tcp://localhost:61616";
		ActiveMQConnectionFactory connectionFactory = new
		ActiveMQConnectionFactory(url);
		CamelContext context = new DefaultCamelContext();
		context.addComponent("jms",
		JmsComponent.jmsComponentAutoAcknowledge(connectionFactory));
		ProducerTemplate producer = context.createProducerTemplate();
		String endpoint =
		"jms:queue:interform.xmlProcessingRequestQueue?jmsKeyFormatStrategy=pas
		sthrough";
		String body = "<xml/>";           // or get input data from an XML-file
		Map<String, Object> headers = new HashMap<String, Object>() {
		   {
		        put("x-interform-metadata-mediaType", "archive");
		        // archive, email or print
		        put("x-interform-metadata-documentType", "postscript");
		        // all the usual metadata for archive, email and print
		        put("x-interform-metadata-com.interform400.xml.Template",
		        "Pangrams.itpl");             // which template to use
		        put("x-interform-metadata-xmlFromSocket",
		        "__XML_FROM_SOCKET__");
		        // skip final step - only create artifact, but don't save, mail
		        or print
		        put("x-interform-metadata-interform.printjob.processingpriority"
		        , "2");
		        // 0=Immediate, 1=Normal, 2=Batch
		        put("x-interform-metadata-interform.plugin.integration.folderNam
		        e", "C:/ProgramData/InterFormNG/temp2");
		        // where to store temporary files
		   }
		};
		byte[] result = (byte[]) producer.requestBodyAndHeaders(endpoint,
		body.getBytes(), headers);
		// the generated artifact is returned

		File pdfFile = File.createTempFile("MQDemo", ".pdf");

		// change extension to match MIME-type of the returned artifact
		OutputStream os = new FileOutputStream(pdfFile);
		os.write(result);
		os.close();
		Runtime.getRuntime().exec(new String[] { "rundll32",
		"url.dll,FileProtocolHandler", pdfFile.getAbsolutePath() });

		System.exit(0);
	}

}
```

Index

Archiving
    Create index xml file. 74
Background image. 22
Configuration
    Users. 29
Conversion
    of templates. 165
Current
    Date. 72
    Time. 72
Data types
    Boolean. 73
    Date. 73
    Number. 73
Date
    Format. 61
Design elements
    Append. 88
    Barcode. 76
    Box. 76
    ColumnBreak. 82
    Define Cursor. 87
    Fixed area. 79
    Flow area. 81
    ForEach. 86
    Group. 79
    If. 84
    Image. 77
    Include. 79
    Label. 78
    Line. 76
    Link. 82
    New Line. 82
    Offset. 87
    Offset update. 88
    Page. 77
    PageBreak. 82
    Repeat. 86
    Select Cursor. 87
    Text. 71
    Variable. 88
    Variable update. 88
Digital signature. 156
Embed files in PDF. 82
Error
    Element unreachable. 97
Fixed Area. 79
Flow Area. 81
Functions. 72
    PAGE_NUMBER. 72
    PAGE_NUMBER_TOTAL. 72
    TIMESTAMP. 72
Getting started. 12
    Setting up the system. 39
If..Else. 85
Image
    Import. 52
    Size. 52
Index xml file. 74
Intro template
    Disable. 41
JD Edwards Connector. 21
Multi language output. 54
Multiup. 103
N-up. 103
New default template. 41
Numeric
    Format. 60
Page number. 72
Pages per side. 103
Password protect PDF. 156
PCL color support. 107
PDF security. 156
PDF Security Module. 21
Printing
    On Zebra printers. 137
    Setup. 127
Reduce the output size. 69
Revision log. 22
Setting up the system. 39
Splitting XML files. 115
Styles
    Barcode style. 64
    Color style. 67
    Error style. 70
    Geometry style. 64
    Label style. 68
    Locale style. 68
    Offset style. 87
    Page style. 67
    Print style. 69
    Text style. 58
Support information. 2
Template
    Conversion. 165
Text to XML. 154
Time stamp. 72
Total number of pages. 72
Transform XML files. 115
Translation tables. 54
Workflow. 109
xml_inbox. 30
Xpath
    Explained. 123
XSLT. 120
Zebra support. 137